March 6 and 7, 1978

Readings - 150

Proceedings
of the

Computer Science and Statistics:

Eleventh Annual Symposium on the INTERFACE

Held at

North Carolina State University

Raleigh, N.C.

Edited by

A. Ronald Gallant
Thomas M. Gerig

Institute of Statistics

North Carolina State University

COMPUTER SCIENCE AND STATISTICS

ELEVENTH ANNUAL SYMPOSIUM ON THE INTERFACE

was supported by grants from

National Science Foundation

Army Research Office, Durham

Office of Naval Research

International Business Machines Corporation

Division of Research Resources, N. I. H.

Division of Computer Research and Technology, N. I. H.

Fogerty International Center, N. I. H.

and held in cooperation with the

Statistical Computing Section of the American Statistical Association

Association for Computing Machinery

North Carolina Chapter of the American Statistical Association

Central Carolina Chapter of the Association for Computing Machinery

ANNOUNCEMENTS

Computer Science and Statistics: Twelfth Annual Symposium on the Interface will be held at the University of Waterloo, Waterloo, Ontario, Canada, on May 10-11, 1979. For information contact Jane F. Gentleman, Department of Statistics, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

Additional copies of these proceedings can be obtained from The Institute of Statistics, North Carolina State University, P. O. Box 5457, Raleigh, NC 27650.

The Proceedings of the Tenth Symposium can be obtained from David Hogben, Statistical Engineering Laboratory, Applied Mathematics Division, National Bureau of Standards, U. S. Department of Commerce, Washington, D.C. 20234.

Proceedings of the Ninth Symposium may be obtained from Prindle, Weber and Schmidt, Inc., 20 Newbury Street, Boston, Massachusetts 02116.

The Proceedings of the Eighth Interface Symposia can be obtained from Health Sciences Computing Facility, AV-111, Center for Health Sciences, University of California, Los Angeles, California 90024.

The Proceedings of the Seventh Interface Symposia can be obtained from Statistical Numerical Analysis and Data Processing Section, 117 Snedecor Hall, Iowa State University, Ames, Iowa 50010.

The Proceedings of the Fourth, Fifth, and Sixth Symposia can be obtained from Western Periodicals Company, 13000 Raymer Street, North Hollywood, California 91605.

CONTENTS

PREFACE

This symposium was the eleventh in a continuing series of symposia organized for those interested in the intersection of the disciplines and professions of computer science and statistics. This series of symposia has, by now, achieved the status of a major national meeting within this community.

The customary format of parallel workshops following a plenary session with keynote address was continued. Each workshop was organized by its chairman and consists of invited speakers presenting papers on the workshop topic followed by open discussion of the paper by all participants. Some workshop chairman have included designated discussants as well. Contributed papers were presented in poster sessions continuing the successful innovation introduced in the ninth symposium.

The major departure from previous symposia is the publication of the proceedings in advance for use by participants during the symposium. Several benefits were anticipated from pre-publication. Much of the distraction of distribution of handouts and note taking during workshop presentations is eliminated. Workshop participants, having read the paper, can eliminate much of the discussion related to clarification of the invited speakers ideas and can focus their comments on the ideas themselves. Parallel workshops unavoidably require the participants to be unable to attend some workshops but, having the papers from these workshops in hand, they may read them and recapture many of the benefits of attendance by discussing the papers with speakers and attendees of the missed workshop during the session breaks, at lunch, and in the evening. Incidentally, one is also assured that invited speakers have adequately prepared their presentations.

Pre-publication entails a rigid and compressed publication schedule. The task of arranging so many papers into a camera ready manuscript in such a short time appeared impossible. Mrs. Margaret Rice single-handedly accomplished the impossible. There are no words adequate to express our debt to her.

It is fitting at the end of a decade to review and record the history of the symposium series. Thus, we were fortunate to have Nancy Mann who is with the

Science Center at Rockwell International as our keynote speaker. She is well-known in the profession for her many professional accomplishments and among them is that she was one of the founders of this series of symposia. In her keynote address she recounts the growth of her progeny. There were nine workshops:

1. Statistical Computing Languages. Organized by Lawrence C. Rafsky. Dr. Rafsky is statistician, Chase Manhattan Bank, and is teaching in the Department of Computer Science, Stevens Institute of Technology. His teaching is in numerical analysis and his primary research interests are in the interface.
2. Computing Methods for Variance Components. Organized by William J. Hemmerle. Professor Hemmerle is Chairman of the Department of Computer Science and Experimental Statistics at the University of Rhode Island. He is a recognized authority on statistical computing for linear models (including variance components) and has published extensively on the topic.
3. Computing for Econometrics. Organized by Warren Dent. Professor Dent is with the Institute for Economic Research, The University of Iowa. He is presently editing a special volume on computing in econometrics for the Journal of Econometrics.
4. Graphics. Organized by Ronald K. Lohrding. Dr. Lohrding, a statistician, is the Group Leader of the Energy Systems and Statistics Group in the Energy Division at the Los Alamos Scientific Laboratory in Los Alamos, New Mexico. Dr. Lohrding and his staff have been doing extensive research into graphic displays of statistical information and data.
5. Statistical Computing in Adversary Proceedings. Organized by John S. deCani. Professor deCani is Chairman, Department of Statistics, The Wharton School, University of Pennsylvania. He has much experience in the preparation of statistical testimony for judicial proceedings.
6. Computing for Bayesian Statistical Methods. Organized by Joseph B. Kadane. Professor Kadane is Head of the Department of Statistics at Carnegie-Mellon University. He is well-known for his research in Bayesian statistical theory and methods and is a major participant in the development of a Bayesian regression procedure in the NBER TROLL system.
7. Numerical Algorithms. Organized by William J. Kennedy. Professor Kennedy is Head of the Numerical Analysis Section of the Statistical Laboratory at Iowa State University and Chairman of the Committee for Evaluation of Numerical Algorithms in the Statistical Computing Section of the American Statistical Association. He was general chairman for the seventh symposium.

8. Large Data Files. Organized by Frank M. Stitt. Dr. Stitt is Director of Clinical Sciences, ALZA Research, and is associated with the Health Sciences Computing Facility, University of California at Los Angeles. He has much experience in data base management and is responsible for new therapeutic clinical trials activity at ALZA. His research interest is in the development of information systems in clinical research.

9. Combinatorial Computing in Statistics. Organized by Joseph B. Kadane. Professor Kadane's professional activities are summarized above.

A. Ronald Gallant
Thomas M. Gerig
Symposium Chairmen

KEYNOTE ADDRESS

EVERYTHING YOU ALWAYS WANTED TO KNOW ABOUT THE HISTORY OF COMPUTER SCIENCE AND STATISTICS: ANNUAL SYMPOSIA ON THE INTERFACE--AND MORE

NANCY R. MANN

Science Center, Rockwell International
Thousand Oaks, California 91360

KEYNOTE ADDRESS

EVERYTHING YOU ALWAYS WANTED TO KNOW ABOUT THE HISTORY OF COMPUTER SCIENCE AND STATISTICS: ANNUAL SYMPOSIA ON THE INTERFACE--AND MORE

Nancy R. Mann

Science Center, Rockwell International
Thousand Oaks, California 91360

INTRODUCTION

This morning I'm going to talk about the history of *Computer Science and Statistics: Symposia on the Interface* and related events applying particularly to the American Statistical Association. This is something that I haven't spoken about before in front of a large audience and, in fact, something that I hadn't thought about for a good number of years at the time Bob Monroe phoned and invited me to speak. I'm assuming, since we are beginning what would be the second decade of Interface symposia, except for approximately half-year slippages between the third and fourth and the seventh and eighth meetings, that the conference organizers felt this is an appropriate time to look back to see how it all began and what has happened since its beginning.

I wish that on this occasion I could tell you that reflection upon this history has enabled me to glean some universal truths to pass on to you, such as, for example, those in the three-pronged ecological maxim: *everything is connected to everything else, everything is dynamically changing* and *there's no such thing as a free lunch*. Unfortunately I haven't been able to find any.

Before I started writing, I read over the keynote addresses of Frank Anscombe, Dick Hamming, H. O. Hartley, Martin Wilk, Edwin Kuh, John Rice, and Tony Ralston appearing in the Proceedings of previous Interface symposia. I felt much like Charlie Brown on the occasion when Lucy commented to Charlie and to Linus that she had read that if one stared at clouds, it was possible to see various familiar objects in them. "What do you see in those clouds overhead, Linus?" she asked. Linus responded, "I see in the configuration of that cloud over there the outline of British Honduras. And I see in that second cloud a sculpture made by the most outstanding sculptor in contemporary America. The aesthetics are indescribable and very moving! And then in that third cloud I see Saul who became the great Apostle Paul, standing watching those who were stoning to death the first Christian martyr, Stephen."

"Wow, Linus," said Lucy, "that's fantastic! What do you see, Charlie Brown?"

And Charlie Brown said "Well, I was going to say I see a horsey and a ducky, but I changed my mind."

Since it's a bit too late now to change my mind about delivering this keynote address, I'll begin by noting that I was surprised to discover during my reflection that I was present at a substantial number of significant events in the early part of the history of which I shall speak. Let me begin at the beginning.

THE BIRTH OF INTERFACE I

The person whom we oldtimers all know to be the original guru of the Interface is Arnie Goodman, known to his parents as Arnold Frank Goodman. Arnie spent his undergraduate years here at North Carolina State where he worked two summers with "Curly" Lucas. Following his graduation he moved West to Stanford, where, in 1961 he received his PhD in Statistics under the direction of Herman Chernoff. He then settled down in Southern California to work in industry, where he soon became interested in computers and computer systems.

By the time I met Arnie in 1963, he had become convinced that mankind (personkind?) was hungering, though perhaps unconsciously, for a symposium on the interface of computer science and statistics. Sometime in late 1965 after he took office as President-Elect of the Southern California ASA chapter, he seized upon the opportunity afforded by this office to assemble a symposium committee, and he attempted to galvanize us into action.

The committee included representatives from the ASA Chapter, from a loosely-knit Southern California organization called SPEC (Statistical Program Evaluation Committee), to which Arnie had been delivering his Interface propaganda, and from the Los Angeles Chapter of ACM. Sol Pollack, the Chairman of the ACM chapter, worked near Arnie at the Space Division of Rockwell, and he had been converted early on to Interface consciousness.

The committee activities got off to a slow start because we initially made contact with officials of a Business School of a local university who seemed to view this as an opportunity for a large proportion of their staff to participate and to receive honoraria. Since our working budget was what might euphemistically be called "zero based" and since we wanted more freedom in choosing our format and speakers, we declined the honor and looked elsewhere. Committee member Mitchell Locks contacted his second cousin (or maybe it was his wife's second cousin), a man named Larry Emanuel, who was associated with University of California Extension at UCLA. Larry thought the Interface premise was a good one for UC Extension sponsorship and was more than willing to let us have a free hand in planning the program. So we were off and running, and it turned out to be a pleasant alliance. By the time we got together with UC Extension, Arnie was looking forward to being President of the ASA Chapter and decided that it would be inappropriate for him to continue to chair the symposium committee. For reasons known only to him and his maker, he saw me as his most likely successor and managed to convince me that Interface chairmanship was my destiny.

Following the initiation of my chairmanship, the committee met and set down the purpose of the symposium, the essence of which can be found in Mike Tarter's Preface in the Proceedings of the sixth meeting:

> *To facilitate the design and improvement of computing systems and software by use of statistics.*
>
> *To facilitate the implementation of statistical methods by making the most efficient use of computation machinery.*
>
> *To enhance the combined uses of statistics and computation as adjuncts to the basic and applied sciences.*

(A somewhat more succinct version of this purpose is given in the Preface by David Hoaglin and Roy Welsch in the Proceedings of the ninth meeting.) We made decisions about the date and format of the symposium (one day, February 1, 1967, with no parallel sessions). We also discussed possible topics and decided on two that continue to be addressed (*computer simulation* and *applications on the interface*) and two that have apparently faded into Interface oblivion (*computational linguistics* and *artificial intelligence*). And we decided to have a luncheon speaker.

After some deliberation, we agreed on a title for the meeting. It was to be "Computer Science and Statistics: A Symposium on the Interface." Why we gave computer science top billing, I can't remember. I suppose it sounded more euphonious. The title was selected during the period in which colons appeared somewhere in all of Arnie's titles. In his later titling, he replaced the colons by pairs of dashes. If the interface symposium had been originated this year, it would doubtless be called "Computer Science and Statistics--A Symposium on the Interface."

MORE PLANNING AND THE BIG EVENT

It was following my first committee meeting as chairman that I was struck with what turned out to be one of my all-time great ideas. It occurred to me during an inspired moment that Ed Thorp who had written "Beat the Dealer," the classic on blackjack strategy, had fairly recently joined the faculty at nearby U.C. Irvine. Certainly, his simulation of blackjack hands and subsequent determination of a nearly optimal strategy for play was an application on the interface that was made for an Interface symposium luncheon talk. Since we had no money to offer for honoraria or anything else, for that matter, I decided on a two-stage procedure for contacting Thorp and inviting him to speak. Somehow, since he had authored a best seller, I viewed him differently from the way I viewed ordinary statisticians and the like.

I drove to Pasadena to a meeting of the Society of Industrial and Applied Mathematics (SIAM) where I had arranged to meet a good friend, Bernie Geldbaum, of a mutual good friend. Bernie was at that time Chairman of the U.C. Irvine Mathematics Department with which Thorp was and still is associated. I told Bernie about the symposium, our lack of resources and my plans for the luncheon talk; and he agreed to relay the plan to Thorp and let him know that I would be contacting him. In the end my daughter and I visited Thorp, his wife and two little girls at their home for morning tea one Sunday on our way to Newport Beach. We had a delightful visit, and he agreed to drive up to our meeting from Irvine and describe the events leading to the writing of his book. He later gave me a title for his talk, which didn't seem to me to give any hint of what it concerned; so I changed it to "Adventures on the Interface in Beating the Dealer." I thought this would insure attendance of all blackjack addicts on our mailing list. And as it turned out, there were quite a few.

I asked Will Dixon to introduce Ed and contacted Arthur Samuel, then recently retired from I.B.M. and more recently affiliated with Stanford, and asked him to describe some of his pioneering work in *artificial intelligence*. It was this work of Samuel's incidentally, that had originally inspired Arnie to Interface guruship. By the time the committee met again to choose potential speakers to fill out the program, we had distinguished chairmen for all of the sessions and some potentially exciting presentations on tap. The program included a talk on *the spectral analysis of brain waves* and one on *numerical testing of evolution theories*.

After the program was set, we dealt with problems of publicity, including the symposium announcements. I recall spending a lot of time on the phone with Larry Emanuel, a great deal of it in convincing him to keep the price of attendance at five dollars. (As I recall, I felt that a modest price would encourage attendance by those who didn't want to bother with company paperwork.) Just before the announcements were to be printed, however, we found that a new student ruling would prevent us from holding the meeting at the location we had selected on the UCLA campus. Thus, the Miramar Hotel in Santa Monica was to become the site of the first Interface Symposium, and Larry therefore raised the price of admission to six dollars. Since luncheon was included, it was nevertheless, I felt, a bargain not to be missed.

By the day before the meeting was to be held, about one hundred people had registered; but on the morning of the event, an additional hundred people waited in line at the Miramar Hotel to pay the registration fee. In the line I saw Merv Muller for the first time since a number of years earlier when we had sat together as undergraduates in Paul Hoel's statistics classes at UCLA. He was on his way home to Wisconsin from somewhere in the Southwest. More people arrived later in the morning, hoping to get into the luncheon by registering for the symposium. By then, however, we were cheek to jowl in our assigned room and there was no more space for devotees of the Interface, blackjack aficionados or otherwise.

The crowded conditions added to the general air of excitement, and it seemed clear by early in the day that the symposium was a success. Thorp's presentation ranked in excitement with watching Evel Knievel hurling over 35 station wagons and a Greyhound bus in one fell swoop. The only problem was that we had to terminate the discussion period with a few dozen people on the verge of cardiac arrest because they had no opportunity to ask the questions burning inside of them.

All of the expenses of the meeting, mailing lists, printing of the announcements, coffee, meals, handout materials for the meeting, and cufflinks I bought for the speakers and session chairmen at the last minute were underwritten by U.C. Extension against potential income from registrations. Arnie and I each advanced thirty dollars to pay the air fare and hotel bill of one speaker whose plight was described to us at the meeting by one of his colleagues. After actions by the executive committees of the sponsoring chapters of ASA and ACM, we were (I'm happy to say) reimbursed.

THE EARLY ANNUAL SYMPOSIA

It seemed evident after Interface I that there should be an Interface II so we optimistically began planning for *Computer Science and Statistics: Second Annual Symposium on the Interface*. More than half of the committee for the second was made up of stalwarts left over from the first. Again I was chairman and again there was Arnie, along with statisticians Wally Blischke and Liz King, and once more Bob Margolin represented ACM. The American Society for Information Science joined in with the other sponsoring organizations which again included U.C. Extension. Larry Emanuel agreed this time to provide honoraria for some of the speakers. Larry was, in fact, a most agreeable sort and one of our early assets. It seems strange that the first time I saw him in person, after two years of interaction on the telephone, was at the second meeting. (On the day of the first he had had another commitment.)

The second Interface symposium was held January 25-26, 1968 at the International Hotel, near the Los Angeles airport. The third, chaired by Ed Robison (then of TRW Systems and now at Union College) was held January 30-31, 1969 at the same spot. For both meetings the format remained the same as for Interface I, except for a time-frame expansion to two days. This is in contrast to written information you may have seen describing all of the first three symposia as one-day meetings. We added an additional sponsor, the Los Angeles Chapter of IEEE Systems Science and Cybernetics Group, for the third conference. Otherwise, sponsorship remained unchanged.

The topics treated at the second and third meetings were varied, ranging from *Social Applications and Implications, National Information Systems, Jurimetrics* and *Computer Aids to Statistical Education* to *Random Number Generation, Computer Technology,* and *Information Science.* Two well-known personalities who are no longer with us participated in these two symposia. George Forsythe of Stanford spoke about *numerical algorithms* at the second, and "Curly" Lucas of N.C. State discussed "The Role of Statistics and the Computer in Biomathematics" at the third.

The tradition of scintillating luncheon addresses continued. At the second, Ross Adey of the UCLA Brain Research Institute told us on the first day about "The Use of Computers in the Search for Memory Traces in the Living Brain"; and Frank Proschan, then still at Boeing Research Labs, regaled us on the second with "Some Effects Computer Science Has Had on Statistics." If the latter topic seems to you an unlikely one to evoke chuckles, it's only because you haven't heard Frank Proschan at his best. This was the first of what has turned out to be many times I've introduced Frank. It's a privilege not meant for the faint-hearted.

At lunchtime on the first day of the third meeting, Jack Moshman gave us a "Retrospective View of Projected Election Returns: 1968 Version," and Ed Davis, the recently-retired Los Angeles Chief of Police, who is beginning the process of running for Governor of California, told us on the second day about "Law, Order, Data and Computers." You may be surprised, if you've read about Ed Davis lately, to learn that he has been a part-time faculty member at the University of Southern California.

The fourth symposium, which was chaired by Mitchell Locks and held at U.C. Irvine, marked a departure from the original format. The most radical change, aside from the use of campus facilities, was in the organization of the meeting into workshops, a structure that has been continued since. The workshops for the fourth symposium dealt with *medical statistics and computation, secondary education, hardware/software design and evaluation by statistical methods,* and *computer languages for statisticians.* For the first time there was financial support from a government agency, the Army Research Office, Durham, and for the first time a Proceedings was written and distributed to attendees in microfiche form. For subsequent symposia there have been hard copy Proceedings.

The fourth symposium also saw the demise of the luncheon talk and the birth of the keynote address. There were, in fact, two keynote addresses at that meeting. Frank Anscombe spoke of the need for flexibility in statistical computing and Dick Hamming discussed the response of statisticians to the computer and computer science and the response of computer scientists to statistics. Both responses were discussed in terms of what had been done contrasted with what Hamming believed should have been done. It was this talk by Hamming that introduced the concept of *compumetrics*, the measurement of computing systems, to the Interface. A significant consequence of this presentation was the formation of SICMETRICS, a special-interest group on Compumetrics within the Los Angeles Chapter of ACM.

Hamming gave versions of his talk to a meeting of the Southern California chapter of A.S.A and as the General Methodology Lecture at the 1971 Annual Meeting of ASA in Fort Collins, Colorado. The part of it that I recall most vividly from the various presentations I heard pointed out the waste of talent involved in "filling in chinks" by extending or generalizing already proven theorems in statistics. "If you do this, he said, "you will read the paper, and the referee will read it." He observed that the probability that anyone else will wade through it is small. He also noted that one way to become a famous statistician is to find a method of attacking and solving an important realworld statistical problem. The solution need not be elegant, only useful and easy to apply. He discussed several such problems in computer science.

THE INTERFACE AND ASA

Hamming's talk at the ASA Annual Meeting took place during the birth of the ASA Statistical Computing Section, following submittal to the Board of Directors and Council of a petition by members of the Association.

You may not recall that ASA's first advertised advocacy of the interface of computer science and statistics took place at its Annual meeting in December of 1967 in Washington, D.C. And I'm fairly certain that you're unaware that Arnie Goodman was behind it from the beginning.

It happened as it did because I made a trip to Washington, in the early summer of 1977. Shortly before the event, Arnie wrote to Don Riley, then ASA Executive Director, that I was planning to be in Washington for a few days and would be available for a lunch-time discussion on two of those days. The implication seemed to be that it was an opportunity not to be missed.

I had been a member of ASA only a short time in 1967, was not at all active and had never heard of Don Riley. I'm fairly certain that, likewise, Don Riley had never heard of me. Nevertheless, Don wrote me a letter graciously inviting me to meet him and other ASA representatives inside the ladies' entrance of the Cosmos Club during one of my available lunch times. Since my consciousness of male chauvinism was languishing comfortably, yet to be raised, I entered the ladies' entrance of the Cosmos Club at the appointed time with no more than mild amusement. There I met Don, Ed Bisgyer, then Business Manager and now Managing Director of ASA, and Joe Daly, then Chief Mathematical and Statistical Advisor for the Bureau of the Census.

The main remembrance I have of the early part of our luncheon is that Don and I discovered we were fellow Ohioans and consequently hit it off extremely well. This was not so long a time before Don passed away, and it turned out to be the only time I ever saw or interacted with him. On this occasion, however, he was enthusiastic and ebullient. Arnie had given me an Interface message to deliver and before we finished our coffee, I managed to summon it to mind. I can remember now very little of the details of the message or even the point of it all. I can only recall that as I listened to myself I was terribly impressed by my eloquence, a quality I had never felt I'd exhibited on a single previous occasion. On the contrary, comments about my oral presentations usually indicated a discernable lack of eloquence. I do remember that the finish of this impassioned speech concerned the need for a session on computer science and statistics at future national ASA meetings.

Don, while perhaps not so awed as I by the eloquence of my remarks, was sufficiently moved to pound on the table a couple of times and declare that there would be such a session at the next Annual Meeting and that I should organize and chair it. This would entail Ed Bisgyer's contacting Carl Bennett, the Program Chairman for 1967, and having him provide a timeslot.

While organizing the session seemed appropriate enough to me, I wasn't sure that I should be the person to chair it. I remember checking with Ed because he

seemd not to be particularly carried away by the emotion of the moment. But he assured me that it would be most proper. I can't remember Joe Daly's reaction to the proceedings, but presumably he was in general agreement.

Once everything was cleared with Carl Bennett, I contacted John Tukey to be a speaker for the session (which my title consultant and I called "Computer Science and Statistics: A View of the Bridge"). Thus, even though the session began at 8:30 a.m. and the room we were assigned held about 500 people, we had a capacity audience. The next year at the ASA Annual Meeting in Pittsburgh, there were four sessions on the Interface. A *Committee on Computers in Statistics* had also been organized, with membership including Joe Daly and Will Dixon.

In 1972, just after the *Statistical Computing Section* replaced the ASA *Committee on Computers in Statistics*, Will Dixon, its first Chairman, phoned and asked me to draft its Charter. I thought this sounded about as onerous a task to ask anyone to perform as I could imagine, about as much fun as watching paint dry--and said so. I found, however, that it wasn't too difficult to accomplish by simply selecting pieces I liked from existing Section charters. And, of course, the major part of the Scope of the Section was borrowed from the purpose of the Interface Symposia that we had set down in 1966. You might note that the name *Section on Statistical Computing* really doesn't reflect all aspects of this Scope. However, the name was changed from *Section on Statistics and Computers*, given in my original draft.

THE MOVE TOWARD INTERFACE INDEPENDENCE

In 1971 the Interface conference was held outside of Southern California for the first time. The Executive Committee of the Southern California Chapter of ASA voted to allow Mitchell Locks to organize the fifth annual Interface symposium at Oklahoma State University, where he had become affiliated upon leaving Southern California himself. The next year, Mike Tarter, who had acted as an Associate Chairman when the symposium was held at U.C. Irvine asked the Chapter for permission to host the meeting at U.C. Berkeley, where he had moved from U.C. Irvine. Before the organization of the meeting in Berkeley got well underway, Mike, Arnie and I met with Will Dixon, in his office, and put together a verbal agreement that has given the Interface Symposium an identity of its own, independent of potential sponsors. We decided, with later blessings from the ASA Chapter and all pioneer Interfacers, that there would be a committee made up of ex-chairmen of Interface symposia and headed by the most recent chairman. This committee would have the power to decide who would be given the privilege of organizing a next symposium after the one currently being organized. The agreement extends now to people who may have been unaware of the Interface symposia at the time it was made. Ultimately, it could extend to people who weren't yet born when it was made.

DIRECTION OF THE INTERFACE

Since the time of our meeting in Will Dixon's office, the *Section on Statistical Computing* has been listed among the sponsors of the Interface symposia. One can note, too, a large overlap of recent officers of the Section and recent organizers of the Interface symposia. Following Mike Tarter's sixth symposium at Berkeley came Bill Kennedy's Iowa State meeting and Jim Frane's at UCLA (back in Southern California). David Hoaglin and Roy Welsch chaired the ninth at Harvard and MIT, and David Hogben and Dennis Fife organized the tenth at the Bureau of Standards.

It is notable that the Statistical Computing Section has seemed to remain true to its name and to have become a section devoted almost entirely to statistical computing. The Interface symposia, on the other hand, have generally continued to treat all three aspects of computer science and statistics set down in 1966 in the original purpose. The sixth, seventh, eighth, ninth, and tenth meetings have offered workshops or papers on, for example, *computer system performance evaluation, robust software, random number generation, evaluation of a management information system, high-level program languages and statistics*, and *pattern recognition*. And Tony Ralston's keynote address for the tenth dealt with the mathematizing of computer science.

One could, I believe, describe the published program for the meeting we're presently attending as one for a symposium on *statistical computing*. The topics are many and varied, and some sound fascinating to me. Still, it seems evident that the organizers of this meeting were not brainwashed in the early traditions we oldtimers were taught to hold near and dear. The chairman of symposium number twelve, Jane Gentleman, can feel free to consider this comment an implicit suggestion carried from the guru, Arnold Goodman, by me, Nancy (just plain) Mann.

WORKSHOP 1

COMPUTING LANGUAGES

Chair: Lawrence C. Rafsky, Chase Manhattan Bank

RIGHTS AND RESPONSIBILITIES OF STATISTICAL LANGUAGE USERS
LANGUAGE STANDARDS FOR STATISTICAL COMPUTING

by

John Brode, Cambridge, MA

The fundamental concept presented in this paper is that of the non-procedural language. The user should only be concerned with what needs to be done; the computer should work out the how. Recent advances in computer science (extension of the properties of context free languages to subsets of context sensitive languages, Petri nets, etc.) can be joined with recent work in mathematics (fuzzy sub-sets, theory of categories, etc.) leading to useful new ideas for statistical computing languages. Standards are proposed for statistical computing languages.

A) Introduction

It is the right of every user of a statistical language to ask that such a language be as useful as the current state of computer science will allow. To get this right, it must be the responsibility of each user to know enough about computer science to distinguish what is known or possible from the individual opinions of practitioners of programming.

The interface between computer science/linguistics and statistics has often been lacking in the past. Far too often, this interface has been between a highly trained statistician and a programmer who is, perhaps, experienced in the writing of some computer language but who is seldom trained in computer science. Too many statistical computing languages seem to have been imposed on statisticians by programmers of very narrow acquaintance with computer science.

The purpose of this paper will not be to present a tutorial in any language, be that language implemented or just proposed. (See the Bibliography for a list of references to more than thirty languages of interest.) Rather, I shall try to present in this paper an outline of what is needed by statistical computing and what can be supplied by computer science.

The fundamental concept contained in this paper is that of a non-procedural program or language. "A non-procedural program is a prescription for solving a problem without regard to details of how it is solved." (Leavenworth & Sammet ('74) p. 2)

> "one wants to deal in so far as possible with the behavior which the program and its parts are to exhibit—and to suppress the detail of just how this is accomplished, demanding of course that the how be consistent with (that is, correctly implement) the desired behavior." (Cheatham & Townley ('75) p. 4)

This is what Sammet ('72) has called a problem defining language. (See also Kassels ('77) and Klinkhamer ('72).)

The fundamental approach taken in this paper is that of a precise algebraic language (see Loos ('74)). This is the approach of APL (Iverson ('62)) but while this language is not the answer for every problem, especially those tending to the non-numeric:

> "There is an important lesson to be learned from APL, however, and that is the importance of structural simplicity and an excellent—smooth and comfortable—system surrounding the language." (Cheatham & Townley ('75) p. 10)

I shall try in this paper to extend this simplicity to handle a wider range of problems that may not be deterministic. To do this, we shall need programs that can by themselves deduce the how of solving a problem.

B. Junction of recent advances in linguistic theory and mathematics

a. Recent advances in linguistic theory

Modern linguistic theory starts with the work of Chomsky ('56 & '59). He defined a series of language types of which type 1 (context sensitive) and type 2 (context free) are of interest here. If there is a grammar rule that changes a token 'A' into a grammatical structure 'w', then these two types may be defined as follows:

type 1: $\phi_1 A \phi_2 \rightarrow \phi_1 w \phi_2$, where ϕ_1, ϕ_2 are arbitrary tokens

type 2: $A \rightarrow w$

Linguistic work done on type 2 languages has been more rigorous, since they are more amenable to precise mathematical treatment. Central to this work is the emptiness problem. Essentially, this problem pertains to the proof that the application of the rules of a grammar a finite number of times to an input stream of tokens from the language generated by the grammar, will not produce an empty result. (See Aho & Ullman ('73) for a somewhat general survey; Hopcroft & Ullman ('69), Hoare & Lauer ('74), or Salomaa ('73) for more precise surveys.)

Unfortunately, too much attention has been given to the neatness of these proofs. Chomsky, in his original work, finds that neither type 1 nor type 2 languages are "exactly what is required for the complete reconstruction of immediate constituent analysis." (Chomsky ('59) p. 148)

Furthermore, in some cases, inadequate attention has been paid to the rigorous meaning (in the mathematical sense) of what has been proven. The emptiness problem is solvable for context free grammars (type 2), but is not solvable for context sensitive grammars (type 1). (Th. 14.2 Hopcroft & Ullman ('69) p. 219) Such a proof assures us that the application of the rules of the grammar by an automatic algorithm will, in a finite number of steps, reduce the input stream to a meaningful statement. This theorem, however, does not prove that no context sensitive (and not context free) grammar exists for which the emptiness problem is solvable.

In particular, starting with the work of Knuth ('65), a form of parsing (a reduction of the input stream according to grammatical rules) called LR(k) has been developed which is coterminus with the set of deterministic languages (essentially languages for which the grammatical rules are unambiguous (Th. 12.1 & Prop. 12.1 Salomaa ('73) p 224 & p 228). The set of deterministic languages, however, is not contained in the set of context free languages nor does it contain the set of context free languages.

It can be shown, that some context sensitive languages are deterministic and that, therefore, the LR(k) approach is applicable.[1] (see Walters ('71)) Thus it is not true that LR(1) parsers need be confined to context free languages.

BNF (Backus-Naur Form,[2] see Naur ('63 & '60) and Backus ('59)) is often touted as the best way to write out the formal description of a grammar. BNF is coterminus with the context free grammars. (see Aho & Ullman ('73)) But since it can describe ambiguous[3] grammars, not all BNF describable grammars are LR(1) parsable. On the contrary, some grammars not describable in BNF are LR(1) parsable. (see Aho, Johnson, & Ullman ('75) for a statement on the inadequacy of BNF. Note that Aho & Johnson ('74) seem to be mistaken in asserting that LR(1) parsing is useful only for context free languages)

It is not possible to show whether an arbitrary context free grammar is ambiguous or not. (Th. 14.7 Hopcroft & Ullman ('69) p. 222) Some context free grammars (and, therefore, their BNF descriptions) can be made ambiguous. (see Earley ('75)) Some recent work has been done on the parsing of languages generated by such ambiguous grammars. (see Aho, Johnson, & Ullman ('75); Aho & Johnson ('74); Earley ('75); Sheil ('76); and Wharton ('76)) The approach taken in these works is to apply "disambiguating rules" whenever there is more than one possible interpretation of an input construct.

An example can be taken from operator precedence. Standard FORTRAN precedence is for the operator '*' (multiplication) to take precedence over '+' (addition). In BNF, this is:

<expression>::=<term>|<expression>+<term>

<term>::=<identifier>|<term>*<identifier> [4]

[1] The 'k' in LR(k) refers basically to the number of symbols that, at most, are to be examined on either side of the current symbol before a decision can be made as to the interpretation of that symbol. It can be shown, however, that for any deterministic language, there exists an LR(0) grammar. (See Th. 12.9 Hopcroft & Ullman ('69) p. 185 for the precise statement) For various reasons, LR(0) grammars are most readily handled as LR(1) grammars. (see Aho & Johnson ('74))

[2] BNF is often given as Backus Normal Form instead of Backus-Naur Form. I prefer the latter, reserving the term "normal form" for the most elementary formal structure such as the Chomsky Normal Form or the Greibach Normal Form. BNF is more properly a meta-language—i.e., a language to describe languages.

[3] A grammar is ambiguous if the application of its rules to a word in the language it generates can give two different meanings. (see Salomaa ('73) p. 54 for a more precise definition)

[4]These two lines are to be read as: "an arbitrary expression is defined either as an arbitrary term or as an arbitrary expression 'plus' an arbitrary term" and "an arbitrary term is defined either as an arbitrary identifying symbol or as an arbitrary term 'times' an arbitrary identifying symbol."

This could be written, ambiguously, as:

<expression>::=<identifier>|<expression>+<expression>|<expression>*<expression>

Disambiguation would then follow according to a table of operator precedence.

A language can be deterministic even if it has an ambiguous grammar provided that such ambiguity is resolved (from outside the grammar) in a deterministic fashion.[5]

[5]see Sheil ('76) for a more rigorous exposition of this point. Sheil limits ambiguity away from "direct ambiguity" (i.e., ambiguity that is preserved through reduction).

The disambiguating rules discussed above (essentially a table lookup) are deterministic. Therefore, such a grammar can be LR(1) parsed. (see de Remer ('71))

The importance of ambiguity lies in the virtual impossibility of avoiding it. The languages generated by unambiguous grammars tend to be uninteresting and inflexible. (see Hamlet ('77)) ALGOL 60, probably the most rigorous high level language written (and for that reason replaced in actual use by the less rigorous ALGOL 68) is known to be ambiguous (Wharton ('76)) and, apparently, cannot be precisely expressed in English. (Hamlet ('77) p. 87-8)[6]

[6]ALGOL is not even a context free language due largely to its "semantic" rules for handling declarations and the like. (see Arbib ('69) p. 172) Nor does it have a "phrase structure grammar" be that context sensitive or context free. (see Floy ('62))

Of equal importance, is the practical result that parsers constructed for certain ambiguous grammars are, in some sense, more efficient than those for unambiguous grammars. (Aho & Johnson ('74)) Sheil ('76) shows that "boundedly ambiguous grammars" can have at most a polynomial bound on the number of steps needed in parsing an input stream if use is made of a "well formed substring table" for disambiguating.

In summary, I have tried to show that recent work on parsing favors deterministic languages and that such languages are not necessarily generated by context free grammars. It has been shown that such a language can be made compatible with an ambiguous grammar. Such ambiguous grammars are certainly more interesting than unambiguous ones. Finally, it would appear that certain ambiguous grammars can be more easily parsed than unambiguous ones.

We should return now to consideration of whether the emptiness problem can be solved for some subset of context sensitive grammars. A partial answer is provided by the nature of LR(1) grammars which may be context sensitive. For such grammars, the emptiness problem can be solved. (see Walters ('71))

Some recent works have shown the emptiness problem to be solvable for certain subsets of context sensitive grammars. The technique used in these proofs is to extend the class of context free grammars to cover special subsets of the context sensitive grammars. Cannon ('76) derives context sensitive grammars by the use of "state generators" (i.e., the interpretation of a given input stream will depend on the "environment" it is placed in.) For such context sensitive grammars, the algebraic (i.e., finite) techniques of the context free grammars are valid.

Chen & Hu ('77) prove that a finite state probabilistic automaton can handle a language that is context sensitive and not context free. This corresponds to proving, essentially, that the successful parsing of a non-deterministic context sensitive grammar is possible in a finite number of steps. This represents a wider extension of the context free grammars than that from either LR(1) parsing or the Cannon results.[7]

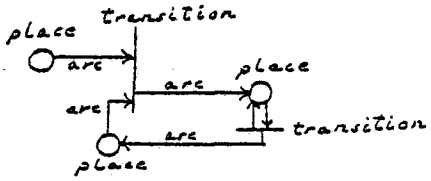
[7]The approach used by Chen & Hu (i.e., limiting the initial states to those having a probability of at least 0.5) is related to that of "level" fuzzy sets proposed by Radecki ('77). An alternate approach to essentially the same problem is to be found in Honda, Nasu, & Hirose ('77) as is described below.

Several authors have shown that divisions can be made between type 2 (context free) and type 1 (context sensitive) languages. Wand ('75) uses algebraic theory to show that type 2 is properly contained in the class of "indexed set" languages which, in turn, are preperly contained in the type 1 languages. Gorun ('76) shows that several proper divisions between types 2 and 1 can be made. Essentially, these new divisions are formed by combining parts or all of context free grammars. These in turn are properly contained in the class of languages that are generated by (handled by) deterministic linear bounded automata (such as LR(1) parsers). It is as yet unproven whether this latter class is equivalent to or is properly contained in the type 1 languages.

Schuler ('74 & '75) has generalized this extension procedure to what he calls "weakly context sensitive" grammars or languages. He defines weakly context sensitive to mean that the context sensitive elements are from a finite number of context free sets. (see Schuler ('74)) Such a language type includes ALGOL 60 thus underlining the virtual non-existence of context free languages of more than trivial

interest. Schuler ('75) warns, however, of the enormous complexity of context sensitivity. He would prefer that grammars stuck to the weakly context sensitive. The extensions above are all in this set of the weakly context sensitive.

The most promising extension of the context free grammars is through the concept of what are called Petri Nets. This concept originated with the work of Petri in the early 1970's. (see Petri ('73 & '75)) A Petri Net is a network relating "places" to each other by means of "transitions" over arcs. Movement from one "place" to another is referred to as "firing" the arc. Such a network can be diagrammed as follows:



Defining a node as either a "place" or a "transition", a Petri Net language is the labeling of the nodes of a Petri Net. Such languages can be classified as follows with regard to the labeling of the nodes (from Peterson ('77)):

1) free — i.e., no two nodes have the same label and no node is without a label.
2) λ-free — i.e., no node is without a label but node labels are not necessarily unique.
3) λ — i.e., some nodes may not have a label and not all nodes have unique labels.

Of these, λ-free is the most interesting sub-set. The set of λ Petri Net languages corresponds to the λ-free set with the unlabeled nodes telescoped into labeled nodes (which could be considered as black boxes from a category as defined below). The set of free Petri Net languages is trivial since every path through a net would be unique and, therefore, not amenable to generalization within the network.

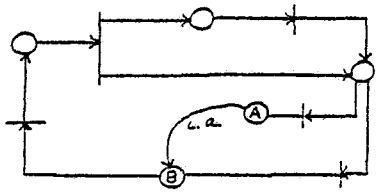
Petri Net languages can be further defined as regards the labels that constitute the terminal state or final markings of the network. (see Peterson ('77)) Of most interest are those Petri Net languages that have labeled terminal states that are, essentially, distinct from the transition states.

It has been shown that all λ-free Petri Net languages are context sensitive. (Hack ('76) or Peterson ('77)) Hack proves that the emptiness problem for terminal Petri Net languages (both λ-free and free) is recursively equivalent to the reachability or finiteness problems. (Th. 9.4 & 9.6 Hack ('76) pp. 150 & 151) He further proves that it is undecidable whether the addition, removal, or changing of any place, transition, arc, or label modifies a Petri net language. (Th. 10.3 Hack ('76) p. 159) In short, it will not be possible to generalize from a specific Petri net language to any subset of these languages defined as an extension of the former.[8]

[8] However, Berthelot & Roucairol ('76) have shown that a Petri net can be reduced in a finite number of steps to an irreducible net with specific properties (the Church-Rosser property, see below). Such a reduction could be used to prove correctness of the nets resulting from a Petri net language. (see Lautenbach & Schmid ('74))

The reachability problem is defined for Petri nets as the question of whether, given an initial set of conditions (or markings) on the Petri net, a terminal state will be reached. The finiteness problem is defined as the question of whether a Petri net will come to a halt after a finite number of firings. Peterson shows that a general algorithm exists for proving reachability in a Petri net language. (Peterson ('77)) As shown in Hack, this will be a recursive problem. (Hack ('76) pp. 150-1)

Peterson briefly touches on generalized Petri nets in which there may be multiple arcs or inhibitor arcs. To wit:



The inhibitor arc 'i.a.' prevents 'B' from firing unless 'A' is set. Such generalizations, while highly desirable for the description of complex networks, are as yet theoretically unmanageable. Nevertheless, as a practical matter, they may be usable. (They relate to both the theory of categories and the theory of fuzzy sub-sets mentioned below.)

The development and extension of Petri Net languages seems capable of releasing the full power of the non-procedural approach to programming. They have the potential of permitting the clear development of a total approach to a problem. In the introduction to the SIGPLAN Symposium on Very High Level Languages, Leavenworth & Sammet define non-procedural to mean that any sequencing needed is done logically by the computer. (Leavenworth & Sammet ('74)) The logic is set by the environment defined in a manner that is, in fact, equivalent to a λ-free Petri Net language. I expect the full development of non-procedural languages to come with that of the Petri Net languages or their equivalent.

As will be made clearer below, a non-procedural approach best suits those problems that require a multi-lateral or parallel attack on the solution. Petri Nets allow a clear definition of both the non-linearity and the multi-linearity of a problem to be made. The computer can then determine the extent of parallel processing that can be used. (see Lautenbach & Schmid ('74))

There remains one further approach to be examined: namely that of pruning dead ends or impossible branches from a network. This involves backtracking from the current node to a prior junction and recursively descending alternate branches. This will be discussed below.

B)b) Recent Advances in Mathematics:

It is not my intent to deal with the advances in mathematics in any detail. The interested reader can pursue these subjects in the references that I shall cite below.

Starting with the work of Zadeh ('65), a great deal has been done on what have come to be called "fuzzy" sets or sub-sets. In somewhat loose terms, a fuzzy sub-set, F, is a set of ordered pairs:

$$F = \left\{ \langle x_i \, \mu_F(x_i) \rangle \right\} \quad \text{for } x_i \in U \;\&\; 0 \leq \mu_F(x_i) \leq 1$$

where x_i is a member of some universal set, U, and $\mu_F(x_i)$ is the measure of membership of x_i in the sub-set F. (see Gaines ('77) for an introduction from the standpoint of logic; Ragade & Gupta ('77) for a moderate mathematical introduction; but especially Kaufmann ('75) for a mathematical text; Gaines & Kohout ('77) is an extensive, annotated bibliography)

The greater part of the work on fuzziness has been algebraic. Topics covered include graphs, logic, networks, as well as extensions to lattices and categories. However, some recent studies have dealt with fuzzy measure theory, thus introducing integration. (Sugeno & Terano ('77)) In particular, this can be extended to feedback systems through the use of convolutions. (Jain ('77))

Fuzziness has been introduced into the rigor of mathematics so as to open new frontiers in our ability to understand the essentially fuzzy world. In the words of Zadeh (forward to Kaufmann ('75) p. ix):

> "We have been slow in coming to the realization that much, perhaps most, of human cognition and interaction with the outside world involves constraints which are not sets in the classical sense, but rather "fuzzy sets" (or subsets), that is, classes with unsharp boundaries in which the transition from membership to non-membership is gradual rather than abrupt."

A practical orientation predominates in the studies of fuzziness as will be seen below. In computer science, fuzziness is most applicable to multi-linear processing where each process is executed with a probability that is less than one.

Starting in the early 1940's, Eilenbery & MacLane ('42) & ('45) proposed what has come to be called the theory of categories. Categories are, essentially, classes of objects that are related to each other through some kind of process.[9]

[9] A precise definition is:

"A Category $\mathcal{C}$ consists of the following data:

i) a class $|\mathcal{C}|$ of objects A,B,C,...

ii) for each ordered pair of objects (A,B) of a (possibly empty) set $[A,B]_{\mathcal{C}}$ called the set of morphisms from A to B,

iii) for each ordered triple (A,B,C) of objects in $\mathcal{C}$, a map $[B,C]_{\mathcal{C}} \times [A,B]_{\mathcal{C}} \longrightarrow [A,C]_{\mathcal{C}}$ called composition of morphisms."

(Schubert ('72) p. 1)

Machines, languages, systems can all be viewed as categories. (A good introduction is MacLane ('72); more rigorous works are Herrlich & Strecher ('73) and Schubert ('72) who is especially clear although quite precise; Makkai & Reyes ('77) cover categorical logic in a strictly mathematical way; Arbib & Manes ('74 & '75) and Goguen ('73 & '76) have done introductory work as well. The reader should be aware, however, that the simpler introductions often lose in clarity what they have gained in simplicity.)

B)c) Application of the junction:

1) Non-procedural Programming:

Leavenworth & Sammet ('74) write of five major features of non-procedural programming languages:

1) associative referencing--i.e., much of what is referred to as relational data base management but extended to automatic referencing (retrieval) of data.
2) aggregate operators--i.e., iterators over sets, matrices etc.
3) elimination of arbitrary sequencing--i.e., mainly that the user supplied input sequence is passed over in favor of logical sequencing.
4) non-deterministic programming and parallel processing.
5) pattern directed structures--i.e., recognition of similar patterns and their replasement by a standard expression or procedure.

(see Kessels ('77) for further discussion of non-procedural programming)

Earley ('74) outlines various basic data types that he feels should be handled in non-procedural programming:

1) tuples--i.e., relational data (e.g., entities, cases, observations, or the like).
2) sequences--i.e., such things as times series and the like that are ordered by their position in the set.

3) sets--i.e., variables, attributes, etc
4) relations--i.e., sets of tuples (a relational data base).
5) functions--i.e., transformations of data or expressions..

Fagin ('76) shows the equivalence of logical implications derived from data and the data base structure of the data. Codd ('72) and McLeod ('77) discuss the language implications of data bases. In summary, it seems important to treat data types and the relations between them with the care and precision of the appropriate algebra.

Earley ('76 & '74) discusses iterators at some length. Iterators can be classified as operators that determine the domain within which an expression or command is to operate. Most commonly, this will be inclusion or exclusion from the set of data to be analysed or otherwise treated. In other words, iterators are functionals that relate one data base to another or one type of data to another (e.g., tuples to sequences). (see Schwarz ('75, '74, & '73) for implementations of iterators in the set oriented language SETL; and Loos ('76) for their implementation in ALDES) This is what Tennent ('76 & '77) refers to as "referential transparancy" by which he means that the program itself derives the functionals necessary to execute the user's commands. Such functionals will determine the environment (domain of action), its content, and its continuation.

Much of what has been said above can be implemented in algebra. A property of algebraic languages is that they are complete.[10] This implies that expressions can be repeatedly reduced without changing their meaning. I.e., if

((A+B)**2) → A**2+2*A*B+B**2

then

(A**2+2*A*B+B**2) → A**2+2*A*B+B**2

This is an useful property in its implication of infinite nesting. However, it is not extendable to the complex domain. E.g.,

$-1=i**2=i*i=\sqrt{-1}*\sqrt{-1}=\sqrt{-1*-1}=\sqrt{1}=1$

Several non-procedural algebraic languages have been written. (see Moses ('71) and Sundblad ('74) for surveys; Kobayashi ('72), Korpela ('76), and Janks ('74) for descriptions of some of these) SCRATCHPAD, besides being non-procedural, is of particular interest in that it is an extensible language. (see Proc Int.Symp on Extensible Lang ('71)) The user can add or modify both its syntax and its semantics--i.e., add and define operators. It also will recognize and reduce some patterns--i.e., reduce 0+x to just x. (see Jenks ('74))

2) Backtracking:

A particularly interesting new application is presented in Stellman & Sussman ('77). They describe a program for the analysis of electronic circuits. Since such circuits can not usually be represented by tractable functions, there is inevitably a certain amount of trial and error in determining the characteristics of a circuit or in finding a circuit with certain characteristics. Their program works by a synchronous search for success or failure on different branches of a logical tree representing the fundamentals of the circuit. The program keeps a trace of its progress and can back track from a failed branch to a higher node that presents an untried branch. (see also Stallings ('76) and Golomb & Baumert ('65) for other examples; but see Montanegro et al. ('77) for an exposition of the authors' preference for heuristic, non-deterministic programming rather than backtracking)

Hamlet ('77) shows that a language accepted by a Turing maching (i.e., type 0 or reccursively ennumerable--see Turing ('36)) will leave a context senstivie trace that will not always be context free. For a simple language (low level), he is able to show that the trace language must not be more restrictive (i.e., of a language with a higher index in the Chomsky hierarchy) than the programming language if the flow of the program is to be properly identified. This would seem to indicate the need for further use of context sensitive trace languages since all high-level languages are context sensitive.

Implicit in backtracking, as well as much of what follows, is parallel processing. (see Grief ('75)) For the moment, this kind of approach is hardware limited. Still, programming languages should be able to produce executable code that is amenable to such processing.

3) Petri Nets:

Barthelot & Roucairol ('76) have shown that Petri Nets have the Church-Rosser property (see footnote 10) under the following rules for reduction:
1) substitution,
2) elimination of redundant places,
3) elimination of irrelevant firing expressions.

Because of the Church-Rosser property, reduction can be used to prove the correctness of parallel processing systems. (see Lautenbach & Schmid ('74))

4) Fuzzy Set Theory:

At the most general level, Santos ('77) has proven the computability on a Turing machine of suitably defined fuzzy programs. He has further shown that a language accepted by a stochastic fuzzy automaton will be regular in the sense of being complete up to the regularity. Honda, Nasu, & Hirose

[10] I.e., that they have the extended Church-Rosser property by which is meant that "any sequence of reductions of an expression 'e' will converge to the same constant 'αe' provided 'αe' is defined", (Loos ('74)) where α is an appropriate factor of similarity or scale.

('77) have proven that a fuzzy language is recognized by an automaton appropriate to the type of language[11] if and only if a cutoff representation of it on a lattice can be recognized by the same automaton.[12] They have further shown essentially, that any 'rational probabilistic event' is recognized by a deterministic linear bounded automaton. (Honda, Nasu, & Hirose ('77) p. 163) A 'rational probabilistic event' is defined as an event realized by some rational probabilistic automaton.[13] This is a further proof of the computability of dependent constructs such as context sensitive languages.

Several applications using fuzzy sets have been made in the planning and control of industrial processes. (see Mamdani ('76) & ('77) for general surveys; and King & Mamdani ('77) for a specific application) Kling ('74) describes the language, PLANNER, which has been modified to deal with fuzzy concepts such as high, low, etc. To date, these applications deal only with those concepts in logical expressions of the form: If the water is low, turn the furnace off.

5) Theory of Categories:

Benson ('75) uses category theory to analyse syntax categories and parsing. He proves that minimal machines[14] can parse context free languages and, which is new, that these machines are simple error detecting. Burstall & Thatcher deal with a form of parallel processing using category theory. Bainbridge ('76) distinguishes linear structures which use linear flowcharts and multi-linear structures which use netWorks He applies category theory to these latter, proving that feedback systems have a minimal representation. Arbib & Manes ('75) show the same for non-deterministic machines.

B)d) Summary:

At one level of linguistic research, there is a movement towards semantics or meaning as opposed to the structure or form of languages. (see Mayne ('75)) Meaning can be fuzzy, categorized, or faulty but needs to be dealt with. (see Hecht ('76)) It has been speculated (Hobbs ('77) that very high level programming languages should follow the lead of natural languages. By this, Hobbs means redundancy, contextual interpretations, inter-sentence relations, and the like. In short, a context sensitive grammar at the least.

At another level, there is a push towards a realization of a more formal logic. "Semantics and realization are aspects of the same situation: semantics is the problem of system analysis; while realization is the problem of system synthesis." (Goguen ('75) p. 151) For Goguen, category theory is the answer since it deals with the formal structure of realization. He would have us work to achieve artificial intelligence defined as "the scientific study (both experimental and theoretical) of the representation, manipulation, utilization, and acquisition of concepts and conceptual systems." (Goguen ('74) p. 547)

Cherniavsky ('78) in a very recent paper would remind us at this point that algorithms and humans can be distinguished by their handling of truth. Humans are always able to adapt to Gödel's "rule of the excluded third". Or, if I may so conclude, to cite Heidegger:

> "Wahrheit ist nicht ursprünglich im Satz beheimatet."
> M. Heidegger, Vom Wesen der Wahrheit, p. 12.
> (Truth is by nature foreign to the sentence.)

---

[11] For a context sensitive language, this will be a linear bounded automaton; for a context free language this will be a push-down automaton.

[12] (see Honda, Nasu, & Hirose ('77) pp. 154-5) A cutoff representation is defined as follows:
1) an L-fuzzy language, f, over Σ is a mapping from Σ^* to L where L is a lattice with minimal element 0, Σ is a finite set of symbols, Σ^* is the set of all strings and symbols in $\Sigma \cup \Lambda$, Λ is the null string. and f(x) for $x \in \Sigma^*$ is the measure of membership.
2) a λ-cutoff representation is the mapping onto L of the set $\{x \in \Sigma^* \mid f(x) \geq \lambda\}$
3) λ is an isolated cutpoint in the sense that every measure f(x) of membership satisfies the following: $|f(x)-\lambda| > \epsilon > 0$.
(see Radecki ('77) who argues on practical grounds for the use of λ-level fuzzy sets for which the measure of membership is set to 0 if it is less than λ)

[13] See an equivalent result in the work of Chen & Hu ('77) mentioned above. Their cutoff point is 0.5 but is used only to show that a language exists that is context sensitive and not context free that is accepted by a finite state probabilistic automaton. Thus, the results of Chen & Hu are not as general as that of Honda, Nasu, & Hirose. Note that the 'rational probabilistic event' above is essentially equivalent to a context sensitive language in the sense that both the interpretation of a context sensitive language and the result of a 'rational probabilistic event' depend on finitely close elements.

[14] Minimal here means, essentially, that there is, at least, a path of a finite number of steps that will accomplish the task.

C) Language Standards for Statistical Computing

a) Introduction:

What are the standards that, by right, the statistician should ask for in statistical computing languages? We have by now covered a considerable amount of ground. What we have seen is, I think, typical of the cutting edge of any discipline--far more questions have been raised than answers returned.

Theory is clearly and naturally further along than the practice of the art. It is precisely into areas where theory has broken new ground and presented new approaches which are of interest to statisticians, that we should insist that programmers venture. Such experimental work will prepare us to match the abilities of computer hardware developments.

I shall outline below a number of standards that, I feel, statisticians should insist be encoded in statistical computing programs. Some of the standards are quite general but some relate to my personal preference for algebra. Without succumbing to the pervasive use of APL, I feel that since written work on the theory of statistics is invariably cast in algebraic notation, a statistical computing language should do likewise. It is my contention that the rigor accompanying such an approach will make the use of such programs in fact easier for the novice than would languages closer to a natural language. (see Francis et al. ('76) for an orientation more towards an English like syntax; Salton ('75) warns that natural language needs to be refined for effective computer use by enhancing the level of disambiguity (sharpen the meaning)).

Following Tucker ('75), I shall first outline the qualities a very high level language should have[15]:

[15] Statistical computing languages are very high level; FORTRAN is high level; and assembly language is low level.

1) easy to learn by a non-programmer,
2) functionally extensible,
3) minimal writing (i.e., abbreviations),
4) permit algorithmic specification,
5) natural to use,
6) portable (to another machine),
7) run-time efficient,
8) capabilities for both batch and interactive,
9) modular,
10) permit access to system functions

(see also Balzar et al. ('76) Point 4 corresponds, more or less, to my interest in an algebraic approach. This conflicts to some extent with point 5.

A further extension of very high level language is to be seen in the work of Kessels ('77) following on that of Klinkhamer ('72). They stress non-proceduralness and contrast this with the presently more prevalent sequential programming. Kessels' non-procedural language deals with blocks of commands. The components of these blocks are then exectued in an order determined by the computer--according to need or logic.

Earley ('74) and Goldsmith ('74) introduce automatic operators over sets (iterators) Datasets would be referenced by association with a name or function. The domain of the set (number of elements, their nature or form, etc.) would be determined by the computer. Operations would then be performed over all of the elements (or some subset thereof) as determined by the environment of the problem.

Loos ('74) shows that an algebraic language can be extended to higher levels of language. Kanoui ('76) describes an implementation of such a higher level algebraic language that is, basically, non-procedural. This language, PROLOG, uses predicate logic over a non-deterministic tree to integrate the symbolic representation of an algebraic expression. (see Sundblad ('74) and Moses ('71) for surveys of algebraic computer languages.)

In the outline below, I shall attempt to apply the four overall tendencies described above in this paper--Non-proceduralness, extended algebra (Petri nets, categories, etc.,), fuzziness, and artificial intelligence. The four are not really separable. Realizations of Petri nets and categories are not likely to be obvious. Therefore, they can not be expressed at once in the proper sequential order (or only inconveniently so) but require that the computer think of how to proceed.

C)b) The obvious--or why do we put up with it?

1) Nothing would be fixed.

There is no need to require that starting in column 16 means something, as is the case in SPSS, for instance.

2) The user should not have to count for the computer.

The computer is capable of counting the number of cases in a dataset or of figuring out the number of variables in a regression.

3) No contrived conventions.

E.g., in IBM JCL, the '=' is used to terminate and begin keyword fields since blanks are used in a very special way (separation of certain major fields). In this case, the computer is not capable of recognizing phrases and needs the contrived convention " keyword = keyword " to recognize one.

4) All programs should be interactive, at least.

A system that runs on batch alone is somewhat comparable to the Model T.

C)c) The not so obvious:

1) Capability for user defined semantics, syntax, and lexics.

All of these should be table driven (i.e., defined by matching with a table) so that it is trivial for a user to change the set of terminal symbols, add a new abbreviation, define a composite operator, or the like.

2) Capability for user defined algorithms.

The program should be sufficiently modularized so that a user could rearrange these modules to create a new algorithm. The user should be required only to specify the new arrangement. The actual rearrangement of machine control should be handled internally by the program.

3) Capability for continuation by means of some other program.

The user should be able to transfer control, data, and whatever else to another program. Again, the actual transfering should be done internally.

4) Abbreviations should be derived algorithmically from the "natural" name.

Thus an user need remember only the full, normal name plus the algorithm, rather than a long series of not always obvious character strings.

C)d) The immediate future or what should be done in the next version:

1) Capability to handle sets and matrices.

Databases should be definable and usable as such (e.g., the database automatically defines the number of entities or attributes to be considered). Matrices should be automatically recognized once defined and handled properly.

2) Capability to operate over sets.

The user should be able to unite or intersect sets; to sub-set by criteria of arbitrary complexity; to relate datasets.

3) Capability to operate over sequences of ordered sets.

The program should distinguish unordered and ordered attributes (such as time series or convergent series) and handle each appropriately. This can range from the trivial level of first differences to the not so trivial ability to loop (iterate) until an arbitrary series converges to some value.

4) Capability to define the environment or to set state descriptors.

The user should be able to set the domain of operations, the extent or form of interaction, output etc. The user should be able to do this on both the global level and on the level of the single operation.

5) Capability to imbed procedures.

Any process that handles information can be treated as a function that may return a single value or multiple values. As such, the user should be able to nest them within other processes or expressions of processes.

6) Capability to recognize patterns.

The program should be able to recognize patterns such as a description of a process and substitute a call to that process for the extended description. The user should be able to specify patterns and to use them as appropriate.

7) Surroundability.

The user should be able to imbed the entire program or just parts of it into some other program or system.

C)e) On the horizon: (all of these concepts are treated more extensively in the text of this paper)

1) Non-proceduralness.

The order of execution of the component parts of a problem should be determined by the computer. This should include the capability of parallel processing where appropriate.

2) Capability of extended algebra.

The user should be able to describe complex multi-linear problems (Petri sets, categorical structures and the like) and expect the program to be able to manipulate and transform them as needed.

3) Inclusion of fuzziness.

The ability to deal with fuzzy attributes as well as fuzzy concepts or networks.

4) Artificial intelligence.

This covers automatic error correction; interactive probing of errors or inconsistencies; determination of how to solve a problem; the ability to use predicate logic; determination of reachability; construction and use of traces for backtracking; etc.

BIBLIOGRAPHY

A.V. Aho, S.C. Johnson, & J.D. Ullman "Deterministic Parsing of Ambiguous Grammars", Communications ACM 18:8 (Aug. 1975) 441-52

A.V. Aho & S.C. Johnson "LR Parsing", Comp Surveys, 6:2 (June 1974) 99-124

A.V. Aho & J.D. Ullman The Theory of Parsing, Translation and Compiling, Englewood Cliffs, N.J.: Prentice-Hall, 1973, 1002 pp. Vol. I: Parsing, Vol. II: Compiling

M.A. Arbib & E.G. Manes "A Category-theoretic Approach to Systems in a Fuzzy World", Synthese, 30: (1975) 381-406

M.A. Arbib & E.G. Manes "Fuzzy Machines in a Category", Bull Australian Math Soc, 13:2 (1975) 169-210.

M.A. Arbib & E.G. Manes Arrows, Structures and Functors: The Categorical Imperative, N.Y.: Academic Press, 1974, 185 pp.

M.A. Arbib & E.G. Manes "Fuzzy Morphisms in Automata Theory" in Proc 1st Int Symp on Category Theory Applied to Computation and Control, San Francisco, 1974; Berlin: Springer Verlag, 1975, 80-6

M.A. Arbib & E.G. Manes "Time Varying Systems", SIAM J. Control, 13:6 (Nov. 1975) 1252-70

M.A. Arbib & E.G. Manes "Machines in a Category: An Expository Introduction", SIAM Rev, 16:2 (April 1974) 163-92

M.A. Arbib
Theories of Abstract Automata, Englewood Cliffs, N.J.:Prentice-Hall, 1969, 412 pp.

E.A. Ashcroft & W.W. Wadge
"LUCID, a Non-procedural Language with Iteration", Communications ACM, 20:7 (July 1977) 519-26

R.R. Avery & C.A. Avery
"Design and Development of an Interactive Statistical System", in Proc Comp Sci & Stat: 8th Annual Symp on the Interface, Los Angeles, CA: Health Sciences Computing Facility, 1975, 49-55

J.W. Backus
Reduction Languages and Variable Free Programming, Yorktown Heights, N.Y.: IBM Res Lab, Report RJ 1010, April 1972

J.W. Backus
"The Syntax and Semantics of the Proposed International Algebraic Language", in Proc Int Conf UNESCO, Paris 1959, Munich: Oldenbourg, 1960

E.S. Bainbridge
"Feedback and Generalized Logic, Info & Control, 31:1 (May 1976) 75-96

R. Balzar, N. Goldman, & D. Wile
"On the Transformational Implementation Approach to Programming", in Proc 2nd Int Comf on Software Eng, 1976, 337-44

D.B. Benson
"An Abstract Machine Theory for Formal Language Parsers", in G. Goos & J. Hartmanis (eds), Category Theory Applied to Computation and Control, Berlin: Springer Verlag, 1975, 106-11

G. Berthelot & G. Roucairol
"Reduction of Petri Nets", in G. Goos & J. Hartmanis (eds), Mathematical Foundations of Computer Science 1976, Berlin: Springer Verlag, 1976, 202-9

R. Bogen
MACSYMA Reference Manual, Cambridge, MA: M.I.T., Project MAC, Nov. 1975, 199 pp.

J. Brode
"Generalizing the Function Call to Statistical Routines--An Application from the DATATRAN Language", paper given at Comp Sci & Stat: 10th Annual Symp on the Interface, 1977

J. Brode, J. Stamen, & R. Wallace
"The DATATRAN Language", in 1976 Proc of the Stat Comp Section, Wash. D.C.: Amer Stat Assoc, 1976, 126-9

J. Brode, P. Werbos, & E. Dunn
TSP/DATATRAN: A Large-scale Statistical Analysis System, Cambridge, MA: M.I.T., Cambridge Project, April 1974, np.

R.M. Burstell & J.W. Thatcher
"The Algebraic Theory of Recursive Program Schemes", in G. Goos & J. Hartmanis (eds), Category Theory Applied to Computation and Control, Berlin: Springer Verlag, 1975, 126-31

R. L. Cannon, Jr.
"An Algebraic Technique for Context-sensitive Parsing", Int J Comp & Inf Sci, 5:3 (Sept. 1976) 257-76

J. M. Chambers
Computational Methods for Data Analysis, N.Y. Wiley, 1977, 268 pp.

T.E. Cheatham, Jr. & J.A. Townley
A Look at Programming and Programming Systems, Cambridge, MA: Harvard Univ, Center for Res in Comp Tech, TR-18-75, 1975, 65 pp.

K.A. Chen & M.K. Hu
"A Finite State Probabilistic Automaton that Accepts a Context Sensitive Language that is not Context Free", Info & Control, 35:3 (Nov. 1977) 196-208

V. Cherniavsky
"On Algorithmic Natural Language Analysis and Understanding", Info Systems, 3:1 (1978) 5-10

D. Chester
"The Translation of Formal Proofs into English", Artificial Intell, 7:3 (Fall 1976) 261-78

N. Chomsky
"Three Models for the Description of Language", PGIT, 2:3 (1956) 113-24

N. Chomsky
"On Certain Formal Properties of Grammars", Info & Control, 2 (1959) 137-67

E.F. Codd
Relational Completeness of Data Base Sublanguages, San Jose CA: IBM Res Lab, Report RJ 987, March 1972

S. Crespi-Reghizzi & D. Mandriolli
"Petri Nets and Szilard Languages", Info & Control, 33:2 (Feb. 1977) 177-92

F.L. de Remer
"Simple LR(k) Grammars", Comm ACM, 14:7 (July 1971) 453-60

J. Earley
"High Level Iterators and a Method for Automatically Designing Data Structure Representation", Comp Lang, 1:4 (1976) 321-42

J. Earley
"Ambiguity and Precedence in Syntax Description", Acta Inf, 4:2 (1975) 183-92

J. Earley
"High Level Operations in Automatic Programming", SIGPLAN Notices, 9:4 (April 1974) 34-42

S. Eilenberg & S. MacLane
"General Theory of Natural Equivalences", Trans Amer Math Soc, 58 (1945) 231-94

S. Eilenberg & S. MacLane
"Group Extensions and Homology", Annals Math, 43 (1942) 757-831

R. Fagin
Relational Database Decomposition and Propositional Logic, San Jose, CA: IBM Res Lab, Report RJ 1776, April 1976, 20 pp.

A.C. Fang
"Generalized Common Sub-expressions in Very High Level Languages", in Conf Record 4th ACM Symp on Principles of Prog Lang, N.Y.: ACM, 1977, 48-57

S.I. Feldman
"A Brief Description of ALTRAN", SIGSAM Bull, 9:4 (Nov. 1975) 12-20

R.W. Floyd
"On the Non-existence of a Phrase Structure Grammar for ALGOL 60", Comm ACM, 5:9 (Sept. 1962) 483-4

I. Francis
"The Statistical Profession and the Quality of Statistical Software", Bull Int Stat Inst, 47 (1977)

I. Francis, R.M. Heiberger, & S.P. Sherman
"Languages and Programs for Tabulating Data from Surveys", in Proc 9th Interface Symp on Comp Sci & Stat, Boston, MA: Prindle, Weber & Schmidt, 1976, 129-39

B.R. Gaines
"Foundations of Fuzzy Reasoning" in M.M. Gupta (ed), Fuzzy Automata and Decision Processes, Amsterdam: North-Holland, 1977, 19-76

B.R. Gaines & L.J. Kohout
"The Fuzzy Decade: A Bibliography of Fuzzy Systems and Closely Related Topics", in M.M. Gupta (ed), Fuzzy Automata and Decision Processes, Amsterdam: North-Holland, 1977, 403-90

J.A. Goguen, Jr.
"Semantics of Computation", in G. Goos & J. Hartmanis (eds), Category Theory Applied to Computation and Control, Berlin: Springer Verlag, 1975, 151-63

J.A. Goguen, Jr.
"Concept Representation in Natural and Artificial Languages: Axioms, Extensions, and Applications for Fuzzy Sets", Int J Man-Machine Studies, 6:5 (Sept. 1974) 513-61

J.A. Goguen, Jr., J.W. Thatcher, E.C. Wagner, & J.B. Wright, A Junction between Computer Science and Category Theory, I, Yorktown Heights, N.Y.: IBM Watson Res Center, Part 1: Report RC 4526, Sept. 1973, 71 pp., Part 2: Report RC 5908, March 1976, 176 pp.

C.W. Goldsmith
"The Design of a Procedureless Programming Language", SIGPLAN Notices, 9:4 (April 1974) 13-24

S. Golomp & L. Baumert
"Backtrack Programming", J ACM, 12:4 (Oct. 1965) 516-24

I. Gorun
"A Hierarchy of Context-sensitive Languages", in G. Goos & J. Hartmanis (eds), Mathematical Foundations of Computer Science 1976, Berlin: Springer Verlag, 1976, 299-303

I. Grief
Semantics of Communicating Parallel Processes, Cambridge, MA: M.I.T., Project MAC, MAC TR 154, Sept. 1975, 161 pp.

M. Hack
Decidability Questions for Petri Nets, Cambridge, MA: M.I.T., Lab for Comp Sci, TR 161, June 1976, 194 pp.

M. Hack
Petri Net Languages, Cambridge, MA: M.I.T. Lab for Comp Sci, TR 159, March 1976, 128 pp.

R.G. Hamlet
"Execution Traces and Programming-language Semantics", Int J Comp & Inf Sci, 6:4 (Dec. 1977) 263-78

R.G. Hamlet
"Syntax and Semantics of Universal Programming Languages", Int J Comp Math, 6:2 Section A (1977) 87-103

M. Hammer, W.G. Howe, V.J. Kruskan, & I. Wladawsky, "A Very High Level Programming Language for Data Processing Applications", Comm ACM, 20:11 (Nov. 1977) 832-40

H. Hecht
"Fault-tolerant Software for Real-time Applications", Comp Surveys, 8:4 (Dec. 1976) 391-407

M. Heidegger
Vom Wesen der Wahrheit, Frankfurt a. M.: Klostermann, 1954, 27 pp.

H. Herrlich & G. Strecher
Category Theory: An Introduction, Rockleigh, N.J.: Allyn & Bacon, 1973, 400 pp.

E. A. Hershy
PSL/II Language Specifications, Ann Arbor, MI: Univ. of Mich, Dept Ind Oper Eng, ISDOS Working Paper 68, Feb. 1973

C.A.R. Hoare & P.E. Lauer
"Consistent and Complementary Formal Theories of Semantics of Programming Languages", Acta Inf, 3:2 (1974) 135-53

J.R. Hobbs
"What the Nature of Natural Language Tells Us about How to Make Natural-language-like Programming More Natural", SIGPLAN Notices, 12:8 (Aug. 1977) 85-93

J. Hopcroft & J. Ullman
Formal Languages and their Relation to Automata, Reading, MA: Addison-Wesley, 1969, 242 pp.

K.E. Iverson
A Programming Language, NY: Wiley, 1962, 286 pp.

R. Jain
"Analysis of Fuzzy Systems", in M.M. Gupta (ed), Fuzzy Automata and Decision Processes, Amsterdam: North-Holland, 1977, 251-68

JANUS Beginner's Manual, Cambridge, MA: M.I.T., Lab Arch & Planning, July 1975, n.p.

JANUS Reference Manual, Cambridge, MA: M.I.T., Lab Arch & Planning, Dec. 1976, 332 pp.

R.D. Jenks
"The SCRATCHPAD Language", SIGSAM Bull, 8:2 (May 1974) 20-30

R.D. Jenks
"The SCRATCHPAD Language", SIGPLAN Notices, 9:4 (April 1974) 101-11

S.C. Johnson
"Compiler-compilers: Where have We been and Where are We Going?", in Proc 9th Interface Symp on Comp Sci & Stat, Boston, MA: Prindle, Weber & Schmidt, 1976, 56-8

H. Kanoui
"Some Aspects of Symbolic Integration Via Predicate Logic Programming", SIGSAM Bull, 10:4 (Nov. 1976) 29-42

E. Kant
"The Selection of Efficient Implementations for a High-level Language", SIGPLAN Notices, 12:8 (Aug. 1977) 140-6

A. Kaufmann
Introduction to the Theory of Fuzzy Subsets: Vol. 1: Fundamental Theoretical Elements, N.Y.: Academic Press, 1975, 416 pp.

J.L.W. Kessels
"A Conceptual Framework for a Non-procedural Programming Language", Comm ACM, 20:12 (Dec. 1977) 906-13

P.J. King & E.H. Mamdani
"The Application of Fuzzy Control Systems to Industrial Processes", in M.M. Gupta (ed), Fuzzy Automata and Decision Processes, Amsterdam: North-Holland, 1977, 321-30

M. Klerer & J. Reinfelds (eds)
Interactive Systems for Experimental Applied Mathematics: Proc ACM Symp, Wash. D.C., Aug. 1967, N.Y.: Academic Press, 1968, 472 pp.

R. Kling
"Fuzzy-PLANNER: Reasoning with Inexact Concepts in a Procedural, Problem-solving Language", J Cybernetics, 4:2 (1974) 105-22
J.F. Klinkhamer
"A Definitional Language", in Proc Online '72 Conf, Uxbridge, Middlesex, U.K.: Brunel Univ, Sept. 1972, 335-53
D.E. Knuth
The Art of Programming, Reading, MA: Addison-Wesley, Vol. I: Fundamental Algorithms, 1975 (2nd ed), 634 pp., Vol. II: Semi-numerical Algorithms, 1969, 623 pp., Vol. III: Sorting and Branching, 1973
D.E. Knuth
"On the Translation of Languages from Left to Right", Info & Control, 8:6 (Oct. 1965) 607-39
I. Kobayashi
"An Algebraic Model of Information Structure and Information Processing", in Proc ACM Nat'l Conf, 1972, 1090-104
J. Korpela
"General Characteristics of the ANALITIK Language", SIGSAM Bull, 10:3 (Aug. 1976) 30-48
P.R. Kosinski
A Data Flow Programming Language, Yorktown Heights, N.Y.: IBM Watson Res Center, Report RC 4264, March 1973, n.p.
Language Structure Group of the CODASYL Development Committee, "An Information Algebra: Phase I: Report, Comm ACM, 5:4 (April 1962) 190-ff
K. Lautenbach & H.A. Schmid
"Use of Petri Nets for Proving Correctness of Concurrent Process Systems", in Int Federation Info Processing Working Conf on Manipulation Lang 1974, Amsterdam: North-Holland, 1974
B.M. Leavenworth & J.E. Sammet
"An Overview of Non-procedural Languages", SIGPLAN Notices, 9:4 (April 1974) 1-12
J.A.N. Lee
Computer Semantics: Studies of Algorithms, Processors, and Languages, N.Y.: Van Nostrand Reinhold, 1972, 397 pp.
S.B. Levenson
Table Producing Language. Version 3.5. User's Guide, Wash. D.C.: Bureau Lab Stat, July 1975
R.G.K. Loos
"The Algorithmic Description Language ALDES (Report)", SIGSAM Bull, 10:1 (Feb. 1976) 15-39
R.G.K. Loos
"Toward a Formal Implementation of Computer Algebra", SIGSAM Bull, 8:3 (Aug. 1974) 9-16
B.P. McCune
"The PSI Program Model Builder: Synthesis of Very High Level Programs", SISPLAN Notices, 12:8 (Aug. 1977) 130-9
S. MacLane
Categories for the Working Mathematician, Berlin: Springer Verlag, 1972, 262 pp.
D.J. McLeod
"High Level Definition of Abstract Domains in a Relational Data Base System", Comp Lang, 2:3 (1977) 61-73
M. Makkai & G.E. Reyes
First Order Categorical Logic, Berlin: Springer Verlag, 1977, 301 pp.
E.H. Mamdani
"Applications of Fuzzy Set Theory to Control Systems: A Survey", in M.M. Gupta (ed), Fuzzy Automata and Decision Processes, Amsterdam: North-Holland, 1977, 77-88
E.H. Mamdani
"Advances in the Linguistic Synthesis of Fuzzy Controllers", Int J Man-Machine Studies, 8:6 (Nov. 1976) 669-78
Z. Manna & R. Waldinger
"Towards Automatic Program Synthesis", Comm ACM, 14:3 (March 1971) 151-65
J.A. Mayne
"Relevance of Computer Science to Linguistics and Vice Versa", Int J Comp & Info Sci, 4:3 (Sept. 1975) 265-79
R.C. Mendelsohn
"The Bureau of Labor Statistics Table Producing Language (TPL)", in Proc ACM 1974 Annual Conf, San Diego, CA. & N.Y.: ACM, 1974, 116-22
A.G. Merten, R.L. Wright, & D. Johnson
"Interface between Data Management Software and Statistical Software", in Proc 9th Interface Symp on Comp Sci & Stat, Boston, MA: Prindle, Weber, & Schmidt, 1976, 140-3
H. Mills
"Software Engineering", Science, 195 (1977) 1199-205
M. Minsky
Semantic Information Processing, Cambridge, MA: M.I.T. Press, 1968, 440 pp.
C. Montanegro, G. Pacini, & F. Turini
"Two-level Control Structure for Non-deterministic Programming", Comm ACM, 20:10 (Oct. 1977) 725-30
D.A. Moon
MACLISP Reference Manual, Cambridge, MA: M.I.T., Project MAC, Dec. 1975, n.p.
J. Moses
"Algebraic Simplification: A Guide for the Perplexed", in Proc 2nd Symp on Symbolic & Algebraic Manipulation, N.Y.: ACM, 1971, 282-304
P. Naur (ed)
"Revised Report on the Algorithmic Language ALGOL 60", Comm ACM, 6:1 (Jan. 1963) 1-23
P. Naur (ed)
"Report on the Algorithmic Language ALGOL 60", Comm ACM, 3:5 (May 1960) 299-314
T.P. Neff & A. Kendel
"Simplification of Fuzzy Switching Functions", Int J Comp & Info Sci, 6:1 (March 1977) 55-70
G. Paun
"On the Index of Grammars and Languages", Info & Control, 35:4 (Dec. 1977) 259-66
P. Penfield, Jr.
MARTHA: User's Manual, 1973 Addendum, Cambridge, MA: M.I.T., Dept Elec Eng, 1974, 98 pp.
P. Penfield, Jr.
MARTHA: User's Manual, Cambridge, MA: M.I.T. Press, 1971, 120 pp.
J.L. Peterson
"Petri Nets", Comp Surveys, 9:3 (Sept. 1977) 223-52

C.A. Petri
Interpretations of Net Theory, Bonn: Gesellschaft für Mathematik und Datenverarbeitung, Interner Bericht 75-07, July 1975, 34 pp.

C.A. Petri
"Concepts of Net Theory", paper presented at Symp on Math Foundations Comp Sci 1973, High Tatras, Czech.

Proceedings of the International Symposium on Extensible Languages, SIGPLAN Notices, 6:12 (Dec. 1971) 147 pp.

A.L. Pugh, III
Dynamo User's Manual, Cambridge, MA: M.I.T. Press, 1976 (8th ed), 131 pp.

T. Radecki
"Level Fuzzy Sets", J Cybernetics, 7:3-4 (July-Dec. 1977) 189-98

R.K. Ragade & M.M. Gupta
"Fuzzy Set Theory: Introduction", in M.M. Gupta (ed), Fuzzy Automata and Decision Processes, Amsterdam: North-Holland, 1977, 105-32

P. Roussel
PROLOG: Manuel d'Utilisation, Marseille: Univ d'Aix-Marseille, UER de Luminy, Groupe d'Intelligence Artificielle, Rapport Interne, Sept. 1975

A. Salomaa
Formal Languages, N.Y.: Academic Press, 1973, 322 pp.

G. Salton
"On the Construction of Effective Vocabularies for Information Retrieval", SIGPLAN Notices, 10:1 (Jan. 1975) 48-55

J.E. Sammet
"An Overview of Programming Languages for Specialized Application Areas", Proc SJCC, 40 (1972) 299-311

E.S. Santos
"Fuzzy and Probabilistic Programs", in M. M. Gupta (ed), Fuzzy Automata and Decision Processes, Amsterdam: North-Holland, 1977, 133-48

E.S. Santos
"Regular Fuzzy Expressions", in M.M. Gupta (ed), Fuzzy Automata and Decision Processes, Amsterdam: North-Holland, 1977, 169-76

H. Schubert
Categories, Berlin: Springer Verlag, 1972, 385 pp.

P.F. Schuler
"WCS--Analysis of the Context-sensitive", Acta Inf, 4:4 (1975) 359-71 & "A Note on Degrees of Context-sensitivity", Acta Inf, 5:4 (1975) 387-94

P.F. Schuler
"Weakly Context-sensitive Languages as Models for Programming Languages", Acta Inf, 3:2 (1974) 155-70

J.T. Schwarz
"Optimization of Very High Level Languages. I: Value Transmission and its Corollaries", Comp Lang, 1:2 (June 1975) 161-94 & "...II: Deducing Relationships of Inclusion and Membership", Comp Lang, 1:3 (Sept. 1975) 197-218

J.T. Schwartz
"Automatic and Semi-automatic Optimization of SETL", SIGPLAN Notices, 9:4 (April 1974) 43-9

J.T. Schwartz
On Programming: An Interim Report on the SETL Project--Installment I: Generalities, N.Y.: N.Y. Univ, Courant Inst Math, Comp Sci Dept, 1973, 160 pp.

M.E. Senko
DIAM II with FLORAL LP: Making Pointed Queries with a Light Pen, Yorktown Heights, N.Y.: IBM Watson Res Center, Report RC 6034, July 1976, 28 pp.

B.A. Sheil
Observations on Context-free Parsing, Cambridge, MA: Harvard Univ, Center Res Comp Tech, TR-12-76, 1976, 34 pp.

Special Issue on AMTRAN
SIGPLAN Notices, 6:11 (Nov. 1971) 94 pp.

W. Stallings
"An Application of Co-routines and Backtracking in Interactive Systems", Int J Comp & Info Sci, 5:4 (Dec. 1976) 303-13

R.M. Stallman & G.J. Sussman
"Forward Reasoning and Dependency-directed Backtracking in a System for Computer-aided Circuit Analysis", Artificial Intelligence, 9:2 (Oct. 1977) 135-96

M. Sugeno & T. Terano
"Analytical Representation of Fuzzy Systems", in M.M. Gupta (ed), Guzzy Automata and Decision Processes, Amsterdam: North-Holland, 1977, 177-90

Y. Sundblad
"Symbolic Mathematics Systems: Now and in the Future", SIGSAM Bull, 8:3 (Aug. 1974) 1-8

R.W. Taylor & R.L. Frank
The CODASYL Database Management Systems, San Jose, CA: IBM Res Lab, Report RJ 1755, March 1975, 98 pp.

R.G. Teitel
"Data Base Concepts for Social Science Computing", in Proc 9th Interface Symp on Comp Sci & Stat, Boston, MA: Prindle, Weber, & Schmidt, 1976, 59-70

R.D. Tennent
"Language Design Methods Based on Semantic Principles", Acta Inf, 8:2 (1977) 97-112

R.D. Tennent
"The Denotational Semantics of Programming Languages", Comm ACM, 19:8 (Aug. 1976) 437-53

A. Tucker
"Very High Level Language Design: A Viewpoint", Comp Lang, 1:1 (Jan. 1975) 3-16

A.M. Turing
"On Computable Numbers with an Application to the Entscheidungsproblem", Proc London Math Soc, 2:42 (1936) 230-65

D.A. Walters
"Deterministic Context-sensitive Languages", Info & Control, 17:1 (Sept. 1971) 14-61

M. Wand
"An Algebraic Formulation of the Chomsky Hierarchy", in G. Goos & J. Hartmanis (eds), Category Theory Applied to Computation and Control, Berlin: Springer Verlag, 1975, 209-13

P.L. Weeks & R.W. Heddinger
"Experience in Using XPL in Constructing TPL (Table Producing Language)", in Proc 9th Interface Symp on Comp Sci & Stat, Boston, MA: Prindle, Weber, & Schmidt, 1976, 54-5

P.L. Weeks
"TPL--A Language for Expressing Complex Table Structures", in Proc 7th Interface Symp on Comp Sci & Stat, Ames, IA: 1973
R.M. Wharton
"Resolution of Ambiguity in Parsing", Acto Inf, 6:4 (1976) 387-95
G.N. Wilkinson
"Language Requirements and Designs to Aid Data Analysis and Computing", Bull Int Stat Inst, 47 (1977)
G.N. Wilkinson
"The Language Approach to Statistical Computing", in Proc 9th Interface Symp on Comp Sci & Stat, Boston, MA: Prindle, Weber, & Schmidt, 1976, 71-3
G.N. Wilkinson
"Some General Criteria for Evaluating Language-controlled Statistical Computing Systems", in Proc 9th Interface Symp on Comp Sci & Stat, Boston, MA: Prindle, Weber, & Schmidt, 1976, 147-8
G.N. Wilkinson & C.E. Rogers
"Symbolic Description of Factorial Models for Analysis of Variance", Applied Stat, 22:3 (1973) 392-9

PROGRAMMING LANGUAGES DESCRIBED IN THE BIBLIOGRAPHY

ALDES: Loos('76)
ALGOL 60: Naur('60&'63)
ALTRAN: Feldman('75)
Moses('71)
AMTRAN: Klerer&Reinfeld('67)
Special Issue('71)
ANALITIK: Korpela('76)
APL: Iverson('62)
BDL: Hammer et al('77)
CHARYBDIS: Klerer&Reinfeld('67)
CODASYL: Kobayashi('72)
Lang Struc Gp('62)
Taylor&Frank('75)
DATATRAN: Brode('74)
DFPL: Kosinski('73)
DIAM II: Senko('76)
DYNAMO: Pugh('76)
EASL: Klerer&Reinfeld ('67)
ELl: Cheathan&Townley ('75)
EXPOUND: Chester('76)
GEDANKEN: Tennent('76)
JANUS: Janus('75&'76)
LIBRA: Kant('77)
MACLISP: Moon('75)
Stallman&Sussman('77)
Sundblad('74)
MACSYMA: Bogen('75)
Moses('71)
MANIAC: Klerer&Reinfeld('67)
MARTHA: Penfield ('71&'73)
MATHLIB: Moses('71)
Sundblad('74)
PLANNER: Kling('74)
PMB: McCune('77)
PML: McCune('77)
POSE: Klerer&Reinfeld('67)
PPL: Goldsmith('74)
PROLOG: Kanoui('76)
Roussel('75)
PSL II: Hershy('73)
REDUCE: Klerer&Reinfeld('67)
Moses('71)
REDUCE2: Sundblad('74)
SAC: Moses('71)
SCRATCHPAD: Jenks('74)
Moses('71)
Sundblad('74)
SETL: Fang('77)
Schwartz('73,'74 &'75)
TPL: Levenson('75)
Mendelsohn('74)
Weeks&Heddinger('71)
Weeks('73)
VENUS: Klerer&Reinfeld ('67)
XPL: Weeks&Heddinger ('71)

Language Design and Statistical Systems

John M. Chambers

Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

The purpose of this paper is to consider the evolution (past and future) of statistical computing systems against the background of some ideas in general language design. The thesis is that some advances in the latter, which could lead to more effective statistical systems, have so far largely been ignored by the designers of such systems. The remarks are based on looking at and using various languages, and particularly on recent experience in designing an interactive language and system at Bell Laboratories. However, the intent is much less to evaluate or advocate individual systems than to consider general developments in the area. Some deliberately strong opinions are presented, in the hope of stimulating discussions.

1. Evolution of General Programming Languages

Early computer codes or programming languages were efforts to cope with a radical new facility. The designers reached for analogies from such diverse sources as formal logic, the wiring of non-programmable calculating machines or the layout of computing procedures ("programs") for human use. An interesting survey of early efforts is given by Knuth and Pardo [6].

As experience was gained and as the underlying computations became faster and larger in scale, a number of general concepts began to appear. These may be usefully grouped under three headings:

1. *syntactic structure*, the expressions which are recognized in the language;
2. *data structure*, the organization of numbers or other forms of information;
3. *semantic structure*, the effect of executing programs in the language.

In each of these areas the specialized work on individual languages has been significantly extended, although not entirely replaced, by general theoretical and applied work. Broadly speaking, the contemporary language designer can and should begin with an understanding of features common to language design and of the reasons for special choices in particular languages. To some extent, the designer can also pick up actual software to provide a start on the language implementation. (Some examples are given below.)

Theoretical and practical advances in each of the three areas have implications for the design of statistical systems, to a greater or lesser extent. In my opinion, the designers of statistical software have largely failed to take advantage of developments in language design. With a few exceptions, statistical systems have been based on ad-hoc and unnecessarily restrictive concepts of language. This is in spite of considerable interest in the form of user language provided. The interest, however, has gone more in the directions of specific phraseology (particularly the issue of using so-called natural language phrases) rather than towards a general syntactic structure. In particular circumstances, a natural-language approach may be useful; for example, it may help to reduce initial trepidation about using a statistical system. However, the overall result has been to make statistical systems more restrictive than they should be. This in turn encourages a "black-box" approach to analyzing data: the user is led to invoke a few high-level operations to produce summaries or fit models, but is often inhibited from looking deeply and critically at the results of these operations. This is ironic since modern computing, particularly in an interactive mode, has far more potential for deep analysis than was previously available. High-speed numerical computation plus graphical displays give the statistician enormous power. The rest of this paper studies some ways in which work in language design can make this power more available.

2. Syntactic Structure

The design of ALGOL60, about twenty years ago, represented a major development in programming languages. One of the most influential aspects was the formal definition of the syntax of the language, using what has come to be called *Backus- Naur Form*. Earlier languages, like FORTRAN, had simply been implemented and then described in a relatively informal way. The formal definition of the syntax in ALGOL60 made considerably more precise what was legal in the language.

Once the syntax has been defined formally, the process of matching a particular piece of user-written program to that syntax becomes mechanical. It is natural then to look for an algorithm which will do this matching, given a suitable description of the language. A number of such algorithms (usually called *compiler-generators* or *compiler-*

compilers) have been developed. The output of such an algorithm is generally another program, which will do the actual analysis of the particular language. Early compiler generators gave substantially less efficient output than a hand-coded compiler for the same language. Later work has resulted in quite efficient compiler-generators for a large class of languages [5]. While the generators are not as portable or as widely available as one would like, they make language design so much more powerful that consideration of them is essential to good system design.

We examine next a few characteristics of syntactic definitions. A fragmentary example will focus our discussion. Suppose we wish to define the class of legal arithmetic expressions, using the operators +, -, *, / and ↑. Let *arith* stand for a general arithmetic expression. All valid instances of *arith* must be produced by a set of alternative rules or *productions*, somewhat as follows (read '|' as meaning 'or' and ':' as meaning 'is').

```
arith:      constant|variable|
            arithoperatorarith|
            '-'arith|(arith);
operator:   '+'|'-'|'*'|'/'|'↑';
```

Everything enclosed in matching quotes is a literal, and all other terms must be defined somewhere in the syntax. Our fragment does not define *constant* or *variable:* for a language like ALGOL60 the latter would be a variable name or subscripted array name.

This simple example illustrates a number of typical features. The definition is *recursive* in that *arith* appears on the right side of its own definition. Without this, all possible rules with two, three, four, etc. operators would have to be described separately. (Some statistical languages, such as BMD [3], still do not allow recursive arithmetic expressions.) Second, notice that some expressions are *ambiguous* according to the rules given. For example, '3 + x * 5' can be matched by two different paths, depending on whether we first match '3 + x' or 'x * 5'. In this case, and with the ordinary meaning for '+' and '*', the expressions is also semantically ambiguous (it gives different numerical values). This and similar ambiguities can be removed by specifying a different binding strength or *precedence* for different operaters. For example, if the language definition specifies that '*' binds tighter than '+' the above expression will always be interpreted as '3+(x*5)', as one presumably intended.

This example suggests that minimal facilities for powerful syntactic definition should include recursive definition with variable precedence. Unnecessarily restrictive syntactic forms, such as the variant on Polish notation used in APL, enforce an unconventional algebraic form which makes programs harder to read and write.

The success of a syntactic definition depends upon subsuming a wide range of useful expressions in a compact set of rules. Otherwise, this is just a rather inefficient way of writing a table of the commands in the language. Similarly, a language which is easy to learn and pleasant to use should have a simple, consistent syntax. In my opinion, many statistical systems are clumsy to use and unattractive

```
CALL MULTR(N,K,XBAR,STDERR,D,RX,RY,ISAVE,
          B,SB,T,ANS)
```

(IBM SSP, 1968)

```
CALL MLREG1(X,N,NDIM,K,Y,B,RESSTD,RSQUAR)
```

(Bell Labs, 19??)

```
FIT COL 4, WEIGHTS IN COL 6,
          VECTORS IN 7,8,9,10
```

(Omnitab, 1968)

```
.
.
REGRESS TITLE IS 'RELATION OF ...'
DEPENDENT IS PRESSURE
END/
```

(BMDP, 19??)

```
.
.
REGRESSION OF Y ON X1 X2 X3
```

(Cooper, 19??)

Figure 1:
Examples of Regression Commands

in appearance because they give each statistical command a separate syntax. A difficulty with a natural language form such as

REGRESS DEPENDENT IS EARNINGS
INDEPENDENT ARE BOOK AND DOW

even when one gets it correct, is that it does not help in learning, say, how to do plotting in the same language.

A better approach, I believe, is to take traditional mathematical notation with operators and function calls, and adapt it to modern interactive computing. Much of the bias against this form arises from the rigidity which algorithmic languages like FORTRAN impose.

For example, Figure 1 shows a regression command, via two FORTRAN subroutines and in several statistical systems. The systems show a wide variety of formats, some more reasonable than others, but all exhibiting a preference for an English-like command rather than a single syntactic form. The main problems with the FORTRAN examples are that the user is forced to provide too much information and that a rigid, positional form of argument list is imposed. All these problems can be alleviated while retaining the power of algebraic notation. The use of self-defining, general data structures eliminates the need for many arguments. Allowing optional arguments and "name=value" format further simplifies the calls. By defining a general data structure (below) one can arrange that all operators and functions return a single result. This opens the way for a very powerful, consistent syntax for statistical operations. Recent experience at Bell Laboratories suggests that both statisticians and others will adapt easily and enthusiastically to this form of language.

3. Data Structure

A general approach to data structures in statistical systems parallels and interacts with the question of general syntax. Programming languages have moved towards a broader and somewhat more unified concept of the permissible data structures. Statistical systems have again lagged in the application of these ideas, largely because we have tried to predefine the set of permitted structures rather than looking for a simple, general grammar on which to build in an open-ended way.

For data structures, the key ability is the creation of *hierarchicial* or *multi-level* data structures [1, Section 3.c]. Given the definition of some *basic* data structures, an arbitrary multilevel data structure is defined as a list of components, each being either a basic structure or another multi-level structure. This concept has been used in one form or another for diverse applications (list processing, business data management, general programming languages). With suitable interpretation, it provides a convenient basis for statistical data structure as well.

Instead of a general approach, however, most statistical systems have attacked data structures piecemeal, or not at all. Looking for common trends, we find three data structures recurring frequently: the data matrix, the multi-way array and the time-series. For example, BMD [3] bases most of its analyses on a data matrix, read from the input stream. APL [4] does all its analysis on multi-way arrays. The TROLL system [7] uses a time-series as its basic structure. Somewhat the exception are languages for survey analysis, which are naturally drawn to a hierarchical data structure (e.g., [2]) but have tended to treat it as a fixed, special type of structure corresponding to collecting data in a hierarchical form.

The problem with an ad-hoc approach is again that it does not help us to grow. While the specific structures mentioned are useful, they are not universal. If no general grammar exists, one either cannot represent more general results at all or is forced to contortions in order to get by. For example, a data matrix is fine for *input* to a regression routine, but does not help much in representing the *output*. Significantly, most statistical systems do not think of the result of a regression as a structure at all. In most cases, there is no regression data structure to represent the output.

General data structures interact with general syntax. If the expressions in a statistical language are to be open-ended and consistent, as suggested in the previous section, the result of an analysis must be a valid expression, both syntactically and as a data structure. Only then can one build new analyses freely, using existing ones.

Without attempting to expand on this idea in detail, a brief example will illustrate the point. Suppose *regress* is a function which returns a hierarchical regression structure. The components of the structure include residuals, coefficients and whatever else is useful for further analysis. Then the following expression does a regression, assigns it the name *myreg* and then extracts the coefficients, called *coef*, and passes them to a function, *abline*, which plots a line from the y = a + bx definition (here the regression line):

abline ((myreg — regress (x,y))$ coef)

We use *expression $ name* to define a component of a structure. The example illustrates the power in a uniform syntax and data structure: expressions, including assignments, can appear essentially anywhere that they make sense, and all functions return a single structure as result. These are ideas which have arisen in various areas of computing, and which have powerful applications in statistical systems.

Finally, we note that a general approach does not preclude having the statistical data structures built into the system. Data matrices, arrays and time-series can be defined easily as multi-level structures (Figure 2).

4. Semantic Structure

Formal syntax describes the legal expressions in a language. Formal semantics defines the meaning of these expressions. It gives a model, often a mathematical model, for the effect of programs in a language. The area of formal semantics is newer, more ambitious and much more difficult than formal syntax. So far, its impact on statistical systems has been effectively nil. This is to be expected, given the currently rather abstract state of most work in the area, and the relative difficulty of most papers on the topic. A good introduction is the recent, readable survey by Sethi[8].

One can foresee, in general if not in detail, the application of formal semantics to statistical languages. One approach to formal semantics uses the concept of the *store*, the currently defined storage locations and their contents [8]. Then the semantics corresponding to expressions in the language syntax, such as discussed in section 2, are built up from formally defined functions, whose arguments are the store and the expression. Given some basic, predefined semantic operations (for example, arithmetic operators like *add)* one can state semantic definitions for expressions. For example, the syntax which allows *expr 1 + expr 2* would be paraileled by the semantic construction which applied the *add* function to the formal value of *expr 1* and *expr 2*, given the current contents of the store.

These ideas, like the syntactic and data structure concepts, should be applicable to statistical languages, with suitable adjustments. For example, the *store* in most models consists of scalar locations in memory. Application to statistical systems suggests that the store would better be the set of currently defined data structures in the data base. The formal definition of *add* would then be generated for some subset of all legal data structures. In fact, we have implemented a similar idea, defining a class of hierarchical structures (called *numerical* structures) to which arithmetic operations can be applied. (See Figure 2.) Formal semantics would make this an abstract, more precise and more portable concept. How to extend the ideas to general analyses, such as regression, is more challenging.

In the long run, there are exciting possibilities here for precise definition of statistical operations. One must expect, however, many years of further work before the possibilities can be realized.

```
vector:          [ name:        NAME ,
                   mode:        LGL | INT | REAL | CHAR ,
                   length:      >0
                   value:       /*pointer to data values*/
                   ]

array:           ( dim:         vector[ DIM , INT , length, value ] ,
                                  /*dimension vector*/
                   data:        vector[ DATA, mode, length, value ] ,
                   ** )

time-series:     (tsp:          vector[ TSP , REAL , 3 , value ] ,
                    /* start , end , periodicity */
                   data:        vector[ DATA, mode, length, value ] .
                   ** )

numeric-
 structure:      ( data:        vector[ DATA , LGL | INT | REAL ,
                                  length , value ] ,
                   ** )
```

Figure 2: Semi-formal Definition
of Data Structures

Notes:

- Square brackets enclose contiguous items.
- Round brackets enclose components of hierarchical structures.
- Capitalized names are literal.
- '**' means any additional components are permitted

References

[1] Chambers, John M., *Computational Methods for Data Analysis*, Wiley, 1977.

[2] Cooper, B. E., Some advances in the design of statistical systems *J. Roy. Stat. Soc. (A)*. 1976.

[3] Dixon, W. J., (editor), *BMDP Biomedical Computer Programs*, Univ. of California Press, 1975.

[4] IBM Corporation, APL Language.

[5] Johnson, S., Compiler-compilers: Where have we been and were are we going? *Proc. Eighth Interface Symposium*, 1975, 56-58.

[6] Knuth, D. E. and Pardo, L. T., The early development of programming languages., *Technical report. STAN-CS-76-562*. Stanford University.

[7] National Bureau of Economic Research, *Troll Reference Manual* NBER Computer Research Center, Cambridge, MA, 1974.

[8] Sethi, R., Semantics of computer programs: Overview of language definition methods. *Bell Laboratories Memorandum*, 1977.

What is a language for statistical computing?

Richard L. Wexelblat
Sperry Univac
Blue Bell, Pa. 19424

ABSTRACT

The computer user is all too often at the mercy of the computer programmer. Lacking knowledge of the full power of what could be done to make the computer easy to use, users have been satisfied with systems that force them to become programmers. Users have accepted inconvenient and hard to learn programs that never quite do what is wanted. The remedy begins with making users aware of what well-designed systems can do. Here is an attempt at such a remedy.

Although this particular paper is directed toward an audience of statisticians and the examples are taken from statistics applications, the moral is general. The person who will eventually use an applications program must be aware of how a properly designed, properly human-engineered system can work and must demand proper behavior, proper documentation, and proper performance.

> Most statisticians realize that if the computer vanished, present-day practical statistics (both practice and theory) would be left high and dry, that we would have to make drastic changes in much of what we do. ...Silently, almost covertly, the computer has become deeply and irretrievably embedded in almost all of statistics. Today this embedding goes on faster and faster.
>
> --- John Tukey*

It didn't take much insight, even in the early days of computers to see that statistics would be greatly affected by computers. Only a few had the insight, however, to see how profoundly statisticians would be influenced by computers. Ideally statistical computation should be part of statistics. In practice it isn't. Trained statisticians should be able to work with computers without formal study of Computer Science. They can't.

We are faced with time-sharing systems where the computer seems to want to share the human's time, rather than vice versa. We possess nicely packaged subroutines that take days to learn to use. We are hosted by operating systems whose primitive operations seem to bear no resemblance to what we want to do. Furthermore, we are afflicted with programming counsellors who, when asked, "how do I do thus-and-so," reply not "here is how," but rather: "why on earth do you want to do that?"

Now I must admit the truth: I am no statistician. I have always thought a Gaussian distribution to be abnormal, a Poisson distribution to refer to placement of sardines in a sandwich. I am a Computer Scientist: I speak FORTRAN like a native; I write PL/I Programs for amusement. What do I have to offer to those interested in languages for Statistical Computing? Perhaps a bit of insight that will help in understanding, in solving some of the problems of working with a computer in a way that is naturally the way statistical problems are normally stated.

In the jargon of my chosen field, what is needed is a very-high-level nonprocedural problem-oriented language. Somewhere close to the end of this paper, I'll give a few hints on what such a language for statistics might look like. First, it will be necessary to define a few terms and give a few examples (and counter-examples).

What is a computer language?

The indulgence of the reader is requested to permit me to start at the lowest level. The basic language of almost all current computers is binary which for convenience is often represented in octal or hexadecimal.

Although the computer can be programmed in its native language (as indeed it was in the bad-old-days), those programmers who still desire to speak to the machine in its native language usually do so through an Assembler. On small (mini- and micro-computers) assembly language is usually translated by a 1:1 mapping onto machine code. On large computers — or those with versatile operating systems, assembly language instructions are often mapped 1:many to machine code.

Programming in machine code is sufficient (but not realistic) for any statistical problem -- or any other computable problem for that matter. Assembly language is sufficient but recommended only to masochists. Although a great deal of assembly coded statistical software has been written in the past, the age of the assembly language programmer has passed. The largest part of applications software is now being written in high-level languages.

The new ANSI Vocabulary on Information Processing (ANSI, 1977) defines a high-level language as a programming language that does not reflect the structure of any one computer or that of any given class of computers. Thus, languages like ALGOL, APL, BASIC, COBOL, FORTRAN, LISP, PL/I, SNOBOL, etc. may be considered high-level. Of course, as is so often the

*Paper, How Computing and Statistics Affect Each Other, prepared for the Baggage Memorial Meeting, London, England, October 18, 1971.

case with lexicon definitions, it misses the point entirely. Machine independence is irrelevant to our concerns. The significance of "high-level" is that one can mean a lot by saying a little. The statement

```
a(i+1) := b(i) + c(i-1)
```

maps onto (i.e. can be translated into) many instructions in machine language. Furthermore, algebraic languages — those designed to express computations - save a good deal of time and effort for the programmer:

- Input and Output - many high level languages provide excellent means for getting data in and results out of the computer.
- Subroutine Libraries - the basic mathematical functions needed in statistical computation are usually available as an intrinsic part of the language.
- Control Structures - many current high level languages (FORTRAN and BASIC being notable exceptions) give the programmer the means to express the sequence of execution of a complex iterative process clearly and concisely.
- Data Structures - Man does not compute on homogenious arrays alone. Many current languages (FORTRAN and BASIC ditto) give the programmer the means to describe the structure of complicated data aggregates.

APL, by the way, is a language that has been recommended for statistics (Rosenkrands, 1974). While APL is splendid for expressing mathematical functions — e.g.

```
(+/OBS)÷⍴OBS
```

is the mean of a set of observations OBS, some formulae can get rather complex as

```
(÷(○2)*÷2)××¯0.5×X×X
```

which is the Gaussian Distribution function. (A reader of a draft of this paper objected to the way Rosenkrands formulated the Gaussian distribution and suggested that

```
((○2)*¯0.5)××¯0.5×X*2
```

would be clearer.)

Statistical Software

If the statistician is (or can hire) a good programmer, everything that needs to be done on the computer can be done in one of the existing high level languages. For many years this is the way it was done — except that many frequently used functions were combined together in packages of subroutines. Sometimes these subroutines were copiously documented, sometimes the user had to take a sketchy description and hope for the best. Appendix 1 contains an example of a rather complete description, a program from the Sperry Univac Stat-Pack (Sperry Univac, 1973).

Extending FORTRAN or PL/I with statistical subroutines, still leaves the problem of having to write, compile, link, and run FORTRAN or PL/I programs. Whether done in batch or on a time-sharing system it's still programming and not statistics.

Some languages — primarily in the Simulation and Modeling applications areas — have gone beyond general purpose languages and included primitives for certain statistical functions. SIMSCRIPT, for example has built-in random number generators for various distributions such as beta, binomial, log normal, normal, etc. For analyzing data, such things as mean, sum of squares, variance, and standard deviation are available. The following are (out of context) examples from SIMSCRIPT (Kiviat, Villanueva, and Markowitz, 1973).

```
SCHEDULE AN ARRIVAL AT TIME.V + EXPONENTIAL.F(MEAN,1)
```

(The first argument of EXPONENTIAL.F is the mean of the distribution; the second is an integer identifying one of a group of independent generators).

```
TALLY M AS THE MEAN AND V AS THE VARIANCE OF X
```

SIMSCRIPT is unlikely to have all of the facilities a statistician will need and sadly, you still have to be a programmer to use SIMSCRIPT. Despite its attempt to look like English, SIMSCRIPT structure uses what has become the most common programming language statement format:

	label	command keyword	operands or options
e.g.			
BASIC	123	LET	A = B+C
FORTRAN	10	CALL	FUNC (A,B,C)
PL/I		PUT	EDIT (A,B,C)(F(6),X(12));

TROLL (NBER, 1975) is a language/system for data analysis, simulation, etc. which uses the classic keyword format although the keywords often bear little resemblance to English (PRTMOD, BINOVAL):

```
REG 1955 TO 1970      (regression on annual data)
REG 1955 1 TO 1970 4 (regression on quarterly data)
```

Using TROLL online, a user can enter and edit data files, enter and edit models and apply models to data. The following is an example of the commands to achieve a time series plot:

```
PERIOD 4 ;
SEARCH SYSLIB-DATA-NBER ;
OUTOPT PLTWIDTH 5 FULEGEND ;
PLTIME 1950 1 TO 1953 1, GC, GDC ;
```

Although the basic operations in TROLL are those needed for data analysis, the notation and syntax are primitive and ungainly in the extreme. And again, although the language used is far from FORTRAN, the user must still worry about writing programs.

One way to remove the programming is to add input and output to the subroutines and let them stand alone as complete programs. Now the user will still have to worry about job control language but the details, the "guts" of the programs, will not be of concern. At least in 1972, the BMD program set from UCLA (Dixon, 1975) was the best known and most widely used "stand alone" package (Schucany, Shannon, and Minton, 1972). Joyce (1972) described BMD as one of the first successful attempts to produce a truly user-oriented system.

> ...A system of well tried and tested batch programs covering an extremely wide range of statistical applications, freely available and written to a consistent set of programming and documentation standards. It can be used by analysts who need to know little more about the mysteries of computing than how to place a job in the batch stream.

Joyce went on to point out, however, that BMD still places the user at the mercy of turnaround time and subject to the vagaries of job control languages.

The next step is to take a BMD-like system and put it on-line. Making the (perhaps rash) assumption that on-line response time is better than batch turnaround, the user can at least improve convenience. Furthermore, where an error in a control card or in data would probably waste an entire batch run, an on-line system would be able to let the user see the error diagnosis immediately and take on-the-spot corrective action.

On-line access to BMD has been achieved in a way by the BMDP program control language that assists in the sequencing of options and data cards. Below is an example of a BMDP "program" adapted from an unpublished memorandum by J. T. Tama (1977).

```
PROBLEM          TITLE = 'ENGINEERING MANPOWER'./
INPUT            VARIABLES = 6.
                 FORMAT = '(2F5.0,F8.0,F10.0,2F7.0)'./
VARIABLE         ADD = 1.
                 NAMES = TIME,DATAENGP,DATATIA,DATATMSA,
                 'DATAG-RE','DATACE/I',LMSA./
PLOT             XVAR = TIME.
                 YVAR = RESIDUAL./
TRANSFORMATION   LMSA = LOG (DATATMSA)./
REGRESSION       DEPENDENT = DATAENGP.
                 INDEPENDENT = 3,5,6,LMSA./
END/             :    (data cards)
                 :
```

Extension from on-line batch to true conversational time-sharing is simple in conception but apparently not in implementation. In 1972 Joyce expressed surprise that there were so few successful interactive statistical systems. Schucany, et al list only 4 (of 37) packages capable of on-line use and of those only 2 appeared to approach a general purpose capability.

Once the programs are on-line, adding conversational interaction is a simple step. Below is a sample dialog adapted from an example in the manual for a Test of Hypothesis series of programs (Sperry Univac, 1975). In the following uppercase is typed by the computer and lower case following a question mark is user response. I have added three parenthetic notes.

```
WOULD YOU LIKE DETAILED INSTRUCTIONS?  no
THE NUMBER OF OBSERVATIONS IS?  12
WOULD YOU LIKE TO SEE THE DATA?  yes
     (tabular listing of data is printed)
IS THE DATA CORRECT?  yes
     (if no, the user could correct bad entries)
THE VALUE OF RHO IS  .925928
S.D. = .301511
...  (some more output)
ALPHA= .10973E-02
THE NULL HYPOTHESIS OF UNRELATED VARIABLES CAN
NOT BE REJECTED FOR ALPHA CHOSEN=<.10973E-02
ANOTHER PROBLEM?  no
BYE
```

It is also possible to combine an entire statistics package into a single interactive program. Appendix 2 contains a sample of an interaction with such a conversational package. Some of the aspects of conversational systems are pointed out in the introduction to the appendix, but a detailed discussion of the human factors aspects of conversational systems is beyond the scope of the present paper.

The Next Step

What then is a language for statistical computing? Based on the discussion above, FORTRAN is...and so are SIMSCRIPT, PL/I, ALGOL, and a host of others. Indeed if we define language for statistical computing as that which one uses to tell the computer to do some statistical computation, then the system job control language, mirabile dictu, is a statistical language! But statisticians should not be afflicted with all the heart-ache and the thousand natural shocks that programmers are heir to.

I seek for the statistician a notation, a language if you will, a way to state the problem that is natural, easy to learn, and easy to use. In particular, I want the user to say what the problem is, not what the method of solution is to be (what, not how). This class of programming language is usually referred to in Computer Science as "very-high level" or "nonprocedural."

In an overview of nonprocedural languages, Leavenworth and Sammet (1974) quote the following example of a "program" in a hypothetical nonprocedural language:

> FIND INTEGERS A, B, C, AND n SUCH
> THAT $n>2$ AND $A^n + B^n = C^n$.

The problem is clearly and completely stated and says what and not how. The following is adapted from an elementary probability text (Parzen, 1960).

> Consider a sequence of independent repeated Bernoulli trials in which the probability of success on any trial is $p = 5/16$. Let Tn be the number of failures encountered before the nth success is achieved. Find E(Tn) and Var(Tn).

Barring problems of ambiguity in English text, there is enough information in both of these examples for an automatic program generator to translate the problem into an executable computer program. It doesn't matter whether the problem is solvable or not.

In a brief paper it is not possible to go deeply into all of the criteria for good programming languages. Let us look at a few examples of existing nonprocedural languages and see what we can learn from them.

Nonprocedural Languages

So far as I have been able to discover, there is no description of a very high level language for statistics in the literature. Perhaps the diversity of what statisticians do is overwhelming. In some areas where the problem space is more restricted, nonprocedural languages permit a great deal to be done with very little said. Consider linear programming. Given the following problem:

> A production manager is considering the manufacture of three products each of which needs work on a lathe, a grinder, and a drill. Product 2 will sell a maximum of 25 units and it is assumed that every unit of products 1 and 3 can be sold. Profits are \$17.00, \$6.90, and \$7.50 respectively. Available machine time and machine hours are

	avl	time for P1	P2	P3
Lathe	90	3.5	4.6	-
Grinder	50	4.3	1.0	2.8
Drill	160	7.5	2.0	3.0

The "program" in UHELP (University of Houston Easy Linear Programming, Donaghey, Dewan, and Singh, 1970) is

```
LATHE:     3.5 P1 + 4.6 P2           <90;
GRINDER:   4.3 P1 + 1.0 P2 + 2.8 P3  <50;
DRILL:     7.5 P1 + 2.0 P2 + 3.0 P3  <160;
SALES:     P2 < 25;
OBJECTIVE PROFIT:  17 P1 + 6.9 P2 + 7.5 P3;
MAXIMIZE:  PROFIT;
```

UHELP permits the user to specify constraints and objective functions and then to maximize or minimize an objective function. After seeing the solution it is possible to add, delete, or change any equation and to try again.

Conversational systems can be carried beyond the Yes-No-numerics level to provide high level systems. Here is a sample dialog from a simulation generation system: the user talks with the computer, defining a simulation problem. The result of the dialog is the generation of a GPSS program that can then be executed (Heidorn 1974). In the dialog, user input is lower case while computer responses are upper case.

When a vehicle arrives at a station, it leaves there immediately if the length of the line at a pump in the station is not less than 2.

75 percent of the vehicles are cars and a fourth are trucks.

There is just 1 pump.

A stimulation run time of 8 hours is desired.

Ask questions for further info.

HOW OFTEN DO THE VEHICLES ARRIVE AT THE STATION?

The arrivals of vehicles are normally distributed with a mean of 8 minutes.

WHAT IS THE STANDARD DEVIATION OF THE NORMAL DISTRIBUTION?

1 minute.

OTHERWISE, WHAT DO THE VEHICLES DO?

What?

AFTER ARRIVING AT THE STATION, IF THE LENGTH OF THE LINE AT THE PUMP IN THE STATION IS NOT LESS THAN 2, THE VEHICLE WILL LEAVE THE STATION. OTHERWISE, WHAT DO THE VEHICLES DO?

They are serviced.

HOW LONG ARE THE VEHICLES SERVICED AT THE PUMP IN THE STATION?

Vehicle service times are exponential.

THE TIME FOR THE VEHICLES TO BE SERVICED AT THE PUMP IN THE STATION IS EXPONENTIALLY DISTRIBUTED. WHAT IS THE MEAN OF THE EXPONENTIAL DISTRIBUTION?

5 minutes for cars and 9 minutes for trucks.

AFTER BEING SERVICED AT THE PUMP IN THE STATION, WHAT DO THE VEHICLES DO?

Leave the station.

WHAT IS THE BASIC TIME UNIT TO BE USED IN THE MODEL?

30 seconds.

THE PROBLEM STATEMENT IS COMPLETE.

How often does a vehicle arrive at the station?

THE TIME BETWEEN ARRIVALS OF THE VEHICLES AT THE STATION IS NORMALLY DISTRIBUTED, WITH A MEAN OF 8 MINUTES AND A STANDARD DEVIATION OF 1 MINUTE.

The standard deviation of the time between arrivals should be 2 minutes.

YES, THE VEHICLES ARE SERVICED AT THE PUMP IN THE STATION.

Those who have not kept up with the state of the art of computer processing of English language text might be surprised by this example. While the problems of complexity and ambiguity of natural language, make it unlikely that full natural language interaction will soon be possible, in areas where the subject matter, vocabulary and syntax are restricted, amazing progress has been made. (Schank, 1977). For a description of some of the problems in using English with computers, see Hill (1972).

Attributes of a Statistics Language

What are criteria for a good language for statistical computing? Perhaps the following:

> Should have all of the common distributions
> Should be able to do hypothesis testing
> Should have analysis of variance
> ...

Well, yes they are, but to my mind far from the most important. Of course the language must have sufficient statistics built in or it's worthless. I'll leave the decision on what is necessary and sufficient to the practicing statistician. My criteria discussed below are not tied to statistics applications; they relate to the general problem of special purpose software for the user who is not a full time programmer.

Criteria for a Good Language

> It is better to know some of the questions
> than all of the answers.
>
> -- James Thurber

Caveat lector: The opinions in this section are my own, based on many years of programming and using computers, teaching and counseling computer users. I have also spent time describing what I feel to be flaws in existing languages (Wexelblat 1976). I do not ask the reader to agree with all of my conclusions, I merely ask that they be read with an open mind. Here are some attributes that I would consider important for a good language and system.

> Accurate and Precise
> Easily accessible
> Easy to learn and use
> Friendly
> Functionally Complete
> Functionally Modular
> Natural
> Understandable
> Well Documented

And here are some explanations, examples, and counter-examples of these attributes.

Accuracy and Precision

For any mathematical application, both accuracy and precision are important. A watch that tells time to the 100th of a second is quite precise. If the watch is 5 minutes off, the accuracy is nil. Precision is the number of decimal places, accuracy is the correctness. I have actually seen a system that used 3.14159 for pi in single precision and 3.14159000000 for pi in double precision.

Accuracy is the responsibility of the implementor and user, precision depends on the hardware (and how it is used). Some computers can do arithmetic all day and never lose a digit precision, some are quite dangerous. A first-time user would do well to check carefully the results of some computations. Of course, the system should provide the means to assist the user to perform the checks.

Ease of Accessibility

How much of the surrounding environment (operating system, control language, file structure, etc.) is the user going to have to learn to use the applications package? Very little, I would hope. Yet there are still systems beautifully self contained in execution that require a user to learn all about the control language, file system, and text editor of the host in order to prepare data. Of course, the implementor is usually at the mercy of the facilities of the host operating system.

Some operating systems are sufficiently oriented toward the non-programmer user (UNIX, for example, Ritchie and Thompson, 1974) that an applications package need not hide the host from the user, while others (names suppressed to protect the guilty) are so inhospitable that it takes days of training just to get started.

Accessibility applies also to operating environment. The best functionality in the world is of no practical use if batch turnaround is 24 hours or if you can't get an access port or if system response time is measured in tens of seconds.

Ease of Learning and Use

Just as one should not have to know all of FORTRAN to be able to write a simple program, a user should not have to learn all of an applications package to use it. Whether a system is easy to learn and use is highly subjective and dependent on the complexity, modularity, and level of documentation as well as on the user's background. Someone who knows statistics should be able to learn and use a statistical subroutine package readily and rapidly.

It is usually within the power of the designer and implementor to make a system hospitable. They should be willing to expend the time and effort to do a good job. The following example of how not to design a language is from GENSTAT a system for statistical analysis (Nelder).

```
'REFERENCE' CIRCLE
'SCALAR' AREA,R
'VALUE' R=3.7
'CALCULATE' AREA = 3.14159*R**2
'PRINT' AREA
'RUN'
'CLOSE' 'STOP'
```

In GENSTAT, all keywords must be in quotes and only the first four letters are used. Thus, the second statement could just as well have been written 'SCALAWAG' AREA,R. Data names in GENSTAT are significant only in the first eight letters, thus ANNUAL-MEAN, ANNUAL-MODE, and ANNUAL-MEDIAN all refer to the same variable.

GENSTAT programs are processed by a translator or interpreter. Putting the keywords in quotes and abbreviating words certainly makes the implementor's job easier -- at the expense of aggravation for the user. If GENSTAT was devised in the last decade (my manual is undated), there is no excuse for such restrictions. As I said earlier, time-sharing is for the human to share the computer's time, not vice-versa.

The language should be reasonable. GENSTAT's default output format is "scientific" notation -- the result of the print statement above is not 43.008, but

```
AREA 4.3008 E1
```

Friendliness

A friend is someone fun to be with, who does things for you and is understanding when you make mistakes. A friendly computer system is one that will detect and perhaps correct mistakes, will detect when you are in trouble and will offer help, will permit you to ask for instructions, and talk to using your own terms.

An outstanding example of friendliness is the DWIM, do-what-I-mean, function of the INTERLISP system (Teitelman, 1974) that assists the user by correcting spelling and syntax errors. The user may disable DWIM, but when enabled, it will catch many errors (e.g. SINE will be "corrected" to SIN -- unless there exists a function called SINE) and let the user know. If DWIM is enabled in "cautious" mode, the user will be shown each error and each suggested correction which may then be accepted or rejected. In "trusting" mode, the user will be informed of corrections after they are made.

Friendly systems don't make you respond 0 or 1 instead of yes or no. They accept Y, YES, Yes, y, and Yes, etc. for yes and N, NO, No, n, and no for no. If a yes or no answer is not yes, they don't assume it was no and go merrily on their way.

Friendly systems are not verbose. They give brief error messages and let you ask for more details. Messages tell what happened to you, not to the program. Not "TABLE OVERFLOW," but rather "TOO MANY OBSERVATIONS."

Friendly systems are not cute or coy. At 3 a.m. after a hard night's computing, is can be extremely exasperating to see "THAT'S A NO-NO. RE-ENTER THE NUMBER AND DO IT RIGHT THIS TIME, DUMMY" for the 123rd time.

Functional Modularity and Completeness

The user should be able to use parts (modules in the computerists jargon) of the system that are logically complete unto themselves. If a system contains Analysis of Variance and Hypothesis Testing it should be possible to do one function without even knowing about the other.

The system should be complete in the sense that anything reasonable to do should be doable. For example, if the Kruskal-Wallace H Test is not a built-in primitive part of the Hypothesis Testing facility then the

user should be able to achieve the Kruskal-Wallace test by combining other functions.

Naturalness

A system is natural to use when you talk to it in the same vocabulary and notation that you use when solving problems without the system -- or when the new notation is a straightforward extension to current notation.

Some people who have learned mathematics feel that mathematical notation is the natural way to express scientific problems. Those who know FORTRAN feel FORTRAN is natural. Others are of the opinion that only English is natural. Lawrence Peter commented that competence, like truth, beauty, and contact lenses, is in the eye of the beholder. Similarly, naturalness is in the mind of the beholder. For example, suppose that A and B are names of vectors of numbers and I wish to set A to the vector of square roots of the members of B and then print A. In PL/I I would say

```
a = sqrt(b);
put (a);
```

The square root function being mathematical in nature uses math notation; the output function is separated. TROLL, the system for data analysis mentioned earlier carries math notation too far. While

```
DO A = SQRT(B);
```

sets A as desired

```
DO A = PRINT (SQRT(B));
```

prints the values at the same time. Thus PRINT is a math function that returns its argument and, as a side effect, prints the argument: in my opinion an operation unnatural to programmer or human alike.

APL, the mathematical programming notation par excellence separates the computation and print functions while still permitting a single statement:

```
⎕←A←B*.5
```

In APL, the rectangle ("quad") is an output function and the asterisk ("star") is the power operation (2*3 is 8 9*.5 is 3).

What is natural for a statistician? That which you, the statistician, can use or learn to use easily and one that permits others to understand what you have written.

Understandability

Programming languages are not only for communication between humans and computers but also for communication among humans. It is necessary that others be able to read and understand what you have written. My primary objections to COBOL and APL are based upon this criterion: COBOL leads to a sense of false complacency and APL seems to encourage a mind-searing complexity. Consider the following example of a program from a proprietary statistical package (SPSS 1976):

```
FILE NAME       DEMO2
VARIABLE LIST   PEOPLE,INCOME,SAVINGS,SALES
INPUT FORMAT    FREEFIELD
N OF CASES      9
INPUT MEDIUM    DISK
VAR LABELS      PEOPLE,POPULATION IN THE LOCALITY IN THOUSANDS/
                INCOME,AVERAGE FAMILY INCOME IN HUNDREDS/
                SAVINGS,PERCENTAGE OF INCOME SAVED?
                SALES,SALES IN THE LOCALITY IN THOUSAND DOLLARS
REGRESSION      VARIABLES = SALES WITH PEOPLE,INCOME,
                SAVINGS(2)
READ INPUT DATA
SAVE FILE
FINISH
```

Despite the somewhat cryptic abbreviations in a few places, chances are very good that someone who understands regression will immediately understand what the above example is trying to do.

Documentation

Documentation should be available on-line. It should contain examples, be clear, complete, and short and to the point.

Conclusion

The problems of designing a language or system for statistical computation are to a large extent the same as the problems of designing languages for any special purpose application.

The designer must concentrate on letting the user say what is to be done without having to go into the details of how it is to be done.

Design is a human endeavor. The ambitious system designer would do well to look first at the user's requirements, next at how to design computer systems, and then at how to design in general. At the end of the reference list, I have included a brief bibliography of six books that are must-reading for the creative designer.

Remember that function and usability are the keys, not flaming originality. The user is a human and the cost of using humans is not on the same scale of using computers. The designer must work as if he or she will be forced to share a very small office with the user.

References

ANSI. 1977. American National Dictionary for Information Processing, X3/TR-1-77. Available from ANSI, 1430 Broadway, New York, NY 10018.

Dixon, W. J., ed. 1975. BMDP: Biomedical Computer Programs. Berkeley: University of California Press.

Donaghey, Charles; Dewan, Prem; and Singh, Darsham. Dec. 1970. A beginner's language for LP. Ind. Eng: 17-24

Heidorn, George E. 1974. English as a very high level language for simulation. Proceedings of a symposium on very high level languages. SIGPLAN Not, 9(4): 91-100.

Hill, I.D. June 1972. Wouldn't it be nice if we could write computer programs in ordinary English - or would it? Comput Bull: 306-12.

Joyce, S. M. 1972. The development of an interactive statistical language. Proc. ONLINE 72, the International Conference on Online Interactive Computing: 477-96. Uxbridge England: Online Computer Systems. Ltd.

Kiviat, P. J.; Villanueva, R.; and Markowitz, H. M. 1973. SIMSCRIPT II.5 PROGRAMMING LANGUAGE, E. C. Russell, ed. Los Angeles CA: Consolidated Analysis Centers Inc.

Leavenworth, Burt M., and Sammet, Jean E. April 1974. An overview of nonprocedural languages. SIGPLAN Not. 9(4):1-12.

NBER. 1975. TROLL primer. Cambridge MA: National Bureau of Economic Research, Inc.

Nelder, J. A. 19?? The GENSTAT Language. GENSTAT Users Guide No. 1. Harpendon Herts England: Statistics Dept., Rothamstead Experimental Station.

Parzen, Emanuel. 1960. Modern Probability Theory and Its Applications: 370. New York: John Wiley & Sons.

Ritchie, Dennis M. and Thompson, Ken. July 1974. The UNIX time sharing system. Common ACM, 17(7): 365-75. (More recent UNIX documentation available from Bell Labs, Murray Hill, NJ 07974)

Rosenkrands, Bent. 1974. APL as a notational language in Statistics. COMPSTAT 74: Proc on Computational Stat: 507-15. Vienna: Physica-Verlag.

Schank, Roger C. Sept. 1977. Research at Yale in natural language processing. SISTM Quarterly 1(1): 5-10.

Schucany, W. R.; Minton. Paul D.; and Shannon, B. Stanley. Jr., June 1972. A survey of statistical packages. Comput Surv. 4(2): 65-79.

Sperry Univac Computer Systems. 1973. Stat-Pack Programmer Reference, Form UP-7502 Rev. 1. Available from Sperry Univac, P. O. Box 500, Blue Bell, Pa. 19424.

Sperry Univac Computer Systems. 1975. VS/9 Application Library, Test of Hypothesis Series, Form UA-0116. Available from Sperry Univac, P. O. Box 500, Blue Bell, Pa. 19424.

SPSS. 1976. SPSS/ONLINE Users Guide. Norwalk CT: National CSS, Inc.

Tama, J. T. 1977, July 11. Introduction to BMDP. Unpublished Technical Memorandum. Holmdel, NJ: Bell Laboratories.

Teitelman, Warren M. 1974. INTERLISP Reference Manual. Palo Alto CA: XEROX Research Center.

Wexelblat, Richard L. 1976. Maxims for malfeasant designers, or How to design languages to make programming as difficult as possible. Proc 2nd Annual Conf on Software Eng: 331-6. Piscataway, NJ: IEEE (Cat. No. 76CH1125-4C).

3.17. SHPCOR - Sheppard's Corrections

3.17.1. Purpose

This subroutine correct moments computed from grouped data using Sheppard's corrections.

3.17.2. User Procedure

3.17.2.1. Entry

CALL SHPCOR(U,N,STEP,NR,$k)

where:

	DESCRIPTION	TYPE
U	is the array of moments from grouped data on input, and the array of corrected moments on output.	singly-dimensioned, floating-point array; input and output
N	the order of the highest moment to be corrected.	FORTRAN integer; input
STEP	the distance between center-points of the groups.	floating-point integer; input
NR	the last moment that was corrected before an overflow was detected.	FORTRAN integer; output
k	is a statement number in the user's program to which control returns in an overflow is detected. (note: the statement number must, be preceded by $ in the calling sequence.)	input

3.17.2.2. Restrictions

$N \geq 2$

3.17.2.3. Special Considerations

The uncorrected moments in the array U on entry are replaced by the corrected moments. Hence, if the user desires to use the uncorrected moments, they must be saved prior to calling SHPCOR.

3.17.2.4. Other Subprograms Required

None.

3.17.2.5. Error Returns

If an overflow is detected by SHPCOR, NR is set equal to the order of the last calculated moment and control is returned to statement k in the calling program.

3.17.3. Supporting Information

3.17.3.1. Mathematical Method

Let f_j be the probability content of the jth group. Let x_j be the centerpoint of the jth interval. Then the grouped rth moment would be $u_r = \sum_j x_j^r f_j$

For any function g(x) we have, using the Euler-Maclaurin summation formula,

$$\int_a^{a+nh} g(x)dx = \left[\sum_{i=0}^{n} g(a+ih) - \tfrac{1}{2}g(0) - \tfrac{1}{2}g(n)\right] - \sum_{j=2}^{m-1} \frac{h^j}{j!} B_j \left[g^{(j-1)}(x)\right]_a^{a+nh} + S_m$$

$$S_m = \frac{nh^m}{m!} B_m g^{(m)}(a + tnh);\ 0 \leq t \leq 1;\ \text{and } m \text{ is even}$$

$$S_m \leq \left| \frac{2nh^m}{m!} B_{m+1}^{(1)}\left(\tfrac{1}{2}\right) g^{(m)}(a + tnh) \right|;\ \leq t \leq 1;\ \text{and } m \text{ is odd}$$

where B_j is the jth Bernoulli number and $B_j^{(1)}(x)$ is the value of Bernoulli polynomial of order n, and degree j at the point x. (See Milne-Thompson, The Calculus of Finite Differences, Chapter 7.5, Macmillan & Co., Ltd., London [1933].) If f(x) is the probability density function from which the data is sampled, assume that

$\frac{d^m f(x)}{dx^m}$ exists and that

$\lim_{x \to \pm\infty} x^r f^{(j)}(x) = 0,\ j = 1,\ldots,m.$

then set

$$g(x) = x^r \int_{-\frac{h}{2}}^{\frac{h}{2}} f(x + t)dt \text{ where h is the stepsize.}$$

Hence g(x) and its first (m + 1) derivatives are continuous with

$$\left[g^{(j)}(x)\right]_{-\infty}^{\infty} = 0,\ j = 0,\ldots,m+1,$$

and using the Euler-Maclaurin expansion, we get

$$\frac{1}{n}\int_{-\infty}^{-\infty}\int_{-\frac{h}{2}}^{\frac{h}{2}} x^r f(x+t)dtdx$$

$$=\sum_j x_j^r \int_{-\frac{h}{2}}^{\frac{h}{2}} f(x_j+t)dt - S_{m+1}$$

$$=\bar{u}_r - S_{m+1}$$

If the multiple integral on the left is absolutely convergent we can write

$$\bar{u}_r - S_{m+1} = \frac{1}{h}\int_{-\infty}^{\infty}\int_{-\frac{h}{2}}^{\frac{h}{2}} (x-t)^r f(x)dtdx$$

$$=\sum_{j=0}^{r/2}\left(\frac{h}{2}\right)^{2j}\binom{r}{2j}u_{r-2j}$$

where u is the corrected kth order moment.

Hence if S_{m+1} is close to zero we equate it to zero and use the above formula recursively to obtain the corrected moments, i.e.,

$$u_r = \bar{u}_r - \sum_{j=1}^{r/2}\left(\frac{h}{2}\right)^{2j}\binom{r}{2j}u_{r-2j}$$

(See Kendall, M.G., Advanced Theory of Statistics, Vol. 1., Griffin, London [1948], Chapter 3.)

3.17.3.2. Programming Method

See Mathematical Method.

3.17.3.3. Storage

SHPCOR	167
OVERFL	57

Total storage 244 positions.

3.17.3.4. Nomenclature

H	is half the stepsize squared.
I	is the argument of the overflow routine.
K	is an index set for a balk.
NJ	is the maximum value of loop index J.
XM	is the value of the correction term subtracted from the uncorrected moment.

All other variables have been defined in paragraph 3.17.2.1.

3.17.4. Test Design

3.17.4.1. Introduction

The test program checks the logical paths and accuracy of subroutine SHPCOR.

3.17.4.2. Comments

The data for these test cases are from M.G. Kendall, Advanced Theory of Statistics, Vol. 1; Griffin, 1948; (chapter 3).

3.17.4.3. Input

Card 1:	Title card. Col. 1-72	12A6 FORMAT 72 columns of alphanumeric information to be printed.
Card 2:	Parameter card. Col. 1-5	I5 FORMAT NT, number of test cases.
Card Set 3:	There will be NT data sets.	
Card 1 of Set 3:	Col. 1-5	I5 FORMAT N, number of moments
Card 2 of Set 3:		8F10.0 FORMAT
	Col. 1-10	STEP, the stepsize
	Col. 11-20	U(1)
	.	. moments
	Col. 71-80	U(7)
Card 3,ff of Set 3:	Col. 1-10	U(8) If there are more than
	11-20	U(9) seven moments, they will
	.	. appear on successive cards,
	.	. eight per card.

A listing of the data deck used by this test program is given below.

```
 TEST OF SHEPPARDS CORRECTION SUBROUTINE
    2
    6
.1        .6666628  .4569655  .3209523  .2303351  .1685129  .1254332
    4
.5        0.        .158524   .104226   .149090
```

3.17.4.4. Test Program Listing

```
      DIMENSION U(100)                                                  TSCORE
      DIMENSION TITLE(12)
 9875 FORMAT(12A6)
 9876 FORMAT(1H1,10X,12A6)
      READ(5,9875) TITLE
      WRITE(6,9876) TITLE
      READ(5,1) NT                                                      TSCORE
    1 FORMAT(I5)                                                        TSCORE
      WRITE(6,2) NT                                                     TSCORE
    2 FORMAT(32H1 TEST FOR SHEPPARDS CORRECTIONS,I5,10HTEST CASES)      TSCORE
    6 FORMAT(18H CORRECTED MOMENTS//(5X,I5,5X,E14.8))                   TSCORE
      DO 11 IT =1,NT                                                    TSCOR5
      READ(5,1) N                                                       TSCOR5
      READ(5,3) STEP,(U(I),I=1,N)                                       TSCORE
    3 FORMAT(8F10.0)                                                    TSCORE
      WRITE(6,4) IT,N,STEP,(U(I),I=1,N)                                 TSCOR5
    4 FORMAT(10H TEST CASE,I5//I5,18H MOMENTS,STEPSIZE=E14.8//(5X,E14.8 TSCORE
     1))                                                                TSCOR5
      CALL SHPCOR(U,N,STEP,NR,$99)                                      TSCORE
      GO TO 11                                                          TSCORE
   99 WRITE(6,5)                                                        TSCORE
    5 FORMAT(/9H OVERFLOW)                                              TSCORE
   11 WRITE(6,6) (I,U(I),I=1,NR)                                        TS3OR5
      STOP                                                              TSCORE
      END
```

Appendix 2 - Interactive Statistics Package: STAT

The following pages are extracted from documentation for a time-sharing system at Bell Laboratories, Holmdel, NJ. The host system, IBM's CPS, is no longer supported there and I believe that the manuals are no longer available.

STAT's conversational facilities were implemented by J. S. Kagle, making use of existing statistical programs. Interestingly , STAT may be invoked either as a conversational interactive program or as a subroutine to another program. In either case, STAT may be instructed to accept commands or data on-line or from a file.

The attached listing gives a summary of STAT's contents and a reminder of how to invoke STAT and identify a data file, followed by a sample interaction. In the interaction, lines typed by the user begin with an underscore; lines typed by STAT do not.

Attempts at including some "friendliness" can be seen at the top of page 112. When the user gives too few data, STAT not only asks for more but tells exactly which are needed. Line 5 on page 112 illustrates what is sometimes called a "fringe" error-- although this particular instance is fairly innocuous.

About half-way down page 112, a frequent problem with conversational programs is illustrated: verbosity. The user, having achieved an elementary statistics (el) listing, now wants to do a T-test (tt) on the same data. It is necessary to answer four questions before being able to say "tt". A more versatile system would have let the user respond tt to the first question, "Another calculation?"

STAT (Statistics Package)

CPS PROGRAM OR CPS SUBROUTINE PROCEDURE

STAT is a package of 16 statistical subroutines:
EL - Elementary Statistics
CO - Correlation Coefficient
SC - Scatter Diagram
HI - Histogram
PL - Plot Multiple Lines
ST - Step-Wise Multiple Regression
RE - Regression (One Independent Variable)
RA - Rank Correlation
PO - Polynomial Regression
CH - Chi Square Test
SI - Simultaneous Equations
FR - Frequency Tabulation
BO - Selection of Observations Below, Between and Above Given Bounds
MO - First 4 Moments
TT - T-Test on Population Means
MU - Multiple Linear Regression

Enter run(STAT)public for the program or

CALL STAT for the conversational subroutine.

The program will request all required information.

Data sets are given dsnames of Haaaa.Bbbbb.dsname,

where aaaa is the user account number, bbbb is the bin number,dsname is the name entered by the user - this dataset will be temporary if the first character is '@'

EXAMPLE:

```
run(STAT)
STAT - General statistics package
Does a file exist containing the data?
no
Enter name for data file
beispi
This program handles a maximum of 10 variables, and 100
observations per variable.

Enter number of observations per variable
10
Enter number of variables
3
Enter 10 observations for variable 1
13 13 14 10 1.8 17
```

```
Enter data items 7 thru 10 separated by blanks or commas.
29,38 17 18.6
Enter 10 observations for variable 2
51 61 71 88 59.87 72 35 87.1 138
Enter data items 10 thru 10 separated by blanks or commas.
10
Enter 10 observations for variable 3
14 16 15.64 35 47 65 41.7 9 15 71
Data listing?
yes
    VARIABLE
OBS        1          2          3
  1      13.0000    51.0000    14.0000
  2      13.0000    61.0000    16.0000
  3      14.0000    71.0000    15.6400
  4      10.0000    88.0000    35.0000
  5       1.8000    59.8700    47.0000
  6      17.0000    72.0000    65.0000
  7      29.0000    35.0000    41.7000
  8      38.0000    87.1000     9.0000
  9      17.0000   138.0000    15.0000
 10      18.6000    10.0000    71.0000
Listing of routines and calling codes?
no
Type code of routine to be used
el
VARIABLE    MEAN    STD.DEV.   STD.ERROR   MAXIMUM   MINIMUM   RANGE
    1      17.140    10.047      3.177      38.000     1.800    36.200
    2      67.297    34.225     10.823     138.000    10.000   128.000
    3      32.934    22.596      7.145      71.000     9.000    62.000
Another calculation?
yes
Same data file?
yes
Data listing?
no
Listing of routines and calling codes?
no
Type code of routine to be used
tt
List of available hypotheses?
yes

OPTION NUMBERS FOR VARIOUS HYPOTHESES AVAILABLE
1   THAT POPULATION MEAN OF B = GIVEN VALUE OF A
2   THAT POPULATION MEAN OF B = POPULATION MEAN OF A
     GIVEN THAT VARIANCE OF B = VARIANCE OF A
3   THAT POPULATION MEAN OF B = POPULATION MEAN OF A
     GIVEN THAT VARIANCE OF B ¬= VARIANCE OF A
4   THAT POPULATION MEAN OF B = POPULATION MEAN OF A
     GIVEN NO INFORMATION ABOUT VARIANCES OF A & B
Enter option number
1
```



```
Enter 2 variables to be tested, in order A,B separated by
blank or comma
3 1
Enter value of A to be used in test
16
     HYPOTHESIS 1     T STATISTICS  .358828     9 DEGREES OF FREEDOM
Another option?
no
Another calculation?
no
STAT terminated.
-
```

The update routine permits the user to increase (to a maximum of 100) or decrease the number of observations per variable, to increase (to a maximum of 10) or decrease the number of variables, to change individual observed values for a variable, to delete variables, and to delete specific observations for every variable.

Suitable answers to the questions posed by the program are "yes", "no" or an integer, as contextually dictated. One exception is the question, "WHICH VARIABLE NEEDS ITS VALUE CHANGED?", to which "none" may be given as a reply.

Note that for multiple deletions, it is best to delete the last variable (or observation) first, since the ordering of variables (or observations) will change after each deletion, e.g. if the 5th and 8th variables are to be removed, delete variable 8 first and then variable 5.

See example below.

```
run(STAT)
STAT - General statistics package
Does a file exist containing the data?
yes
Enter name of data file
beispi
 WANT TO UPDATE DATA?
yes
FILE H1234.B123.BEISPI CONTAINS 3 VARIABLES WITH
10 OBSERVATIONS PER VARIABLE

DO YOU WISH TO CHANGE THE NUMBER OF OBSERVATIONS PER VARIABLE?yes
ENTER THE NUMBER OF OBSERVATIONS PER VARIABLE.8
WHICH OBSERVATION DO YOU WANT TO REMOVE FOR EVERY VARIABLE?2
WHICH OBSERVATION DO YOU WANT TO REMOVE FOR EVERY VARIABLE?2

DO YOU WISH TO CHANGE THE NUMBER OF VARIABLES?yes
```

```
ENTER THE NUMBER OF VARIABLES.2
DO YOU WANT TO DELETE ANY VARIABLE(S) ?yes
WHICH ONE?3


WHICH VARIABLE NEEDS ITS VALUE CHANGED?1

WHICH OBSERVATION NEEDS TO BE CHANGED?8
WHAT IS THE NEW VALUE FOR THIS OBSERVATION?18.6

ARE THERE ANY MORE CHANGES FOR THIS VARIABLE?no

ARE THERE ANY MORE VARIABLES WHICH TO HAVE THEIR VALUES CHANGED?yes

WHICH VARIABLE NEEDS ITS VALUE CHANGED?2

WHICH OBSERVATION NEEDS TO BE CHANGED?7
WHAT IS THE NEW VALUE FOR THIS OBSERVATION?38

ARE THERE ANY MORE CHANGES FOR THIS VARIABLE?no

ARE THERE ANY MORE VARIABLES WHICH NEED TO HAVE THEIR VALUES CHANGED?no

FILE HAS BEEN REWRITTEN
Data listing?
yes
   VARIABLE
OBS          1            2
  1      13.0000      51.0000
  2      10.0000      88.0000
  3       1.8000      59.8700
  4      17.0000      72.0000
  5      29.0000      35.0000
  6      38.0000      87.1000
  7      17.0000      38.0000
  8      18.6000      10.0000
List of routines and calling codes?
no
Type code of routine to be used
-


run(STAT)
STAT - General statistics package
Does a file exist containing the data?
yes
Enter name of data file
beispi
 WANT TO UPDATE DATA?
yes
FILE H1234.B123.BEISPI CONTAINS 2 VARIABLES WITH
8 OBSERVATIONS PER VARIABLE
```



```
DO YOU WISH TO CHANGE THE NUMBER OF OBSERVATIONS PER VARIABLE?no
DO YOU WISH TO CHANGE THE NUMBER OF VARIABLES?yes
ENTER THE NUMBER OF VARIABLES.4
ENTER  8 OBSERVATION(S) FOR VARIABLE 3
15 15.1 16 18 17.9 14 13.9 15 15.1
ENTER 8 OBSERVATION(S) FOR VARIABLE 4
15.1 15.2 15 15.4 1.98 15 16 18

WHICH VARIABLE NEEDS ITS VALUE CHANGED?none
FILE HAS BEEN REWRITTEN
Data listing?
yes
   VARIABLE
OBS          1            2            3            4
  1      13.0000      51.0000      15.0000      15.1000
  2      10.0000      88.0000      15.1000      15.2000
  3       1.8000      59.8700      16.0000      15.0000
  4      17.0000      72.0000      18.0000      15.4000
  5      29.0000      35.0000      17.9000       1.9800
  6      38.0000      87.1000      14.0000      15.0000
  7      17.0000      38.0000      13.9000      16.0000
  8      18.6000      10.0000      15.0000      18.0000
Listing of routines and calling codes?
-
```

WORKSHOP 2

COMPUTING METHODS FOR VARIANCE COMPONENTS

Chair: William J. Hemmerle, University of Rhode Island

A SIMPLE 'SYNTHESIS'-BASED METHOD OF VARIANCE COMPONENT ESTIMATION

by

H. O. Hartley*, J. N. K. Rao+ and Lynn LaMotte#

ABSTRACT

This paper develops a new algorithm for the estimation of components of a variance in the mixed ANOVA model. This algorithm is "efficient" since the computational effort (measured by the number of products) is proportional to n, the number of observations. The method of estimation on which the algorithm is based can be identified with special cases of both MINQUE (for V = I) and with the 1st iterate for the solution of the REML equations. Other optimality properties are established, and simple conditions for estimability of the variance components are derived. The consistency of the estimators is proved, and they are therefore effective starting points for a single cycle of M.L. iterations leading to fully efficient estimates.

*H. O. Hartley, Institute of Statistics, Texas A&M University
+J. N. K. Rao, Carleton University, Ottawa
#Lynn LaMotte, Quantitative Management Science, University of Houston

1. Introduction

Two of us (HOH and JNKR) have recently had occasion (see Hartley and Rao (1977)) to consider components of variance estimation techniques in data banks arising from sample surveys. Such data banks differ from those encountered in experimental designs in that the "number of observations", n (in our case the number of elementary sampling units), is exceedingly large. We have therefore been prompted to search for computationally efficient methods for the estimation of components of variance when n is large and the algorithm here described involves a computational effort (as measured by the number of products) which is a linear function of n and this is generally regarded as computationally highly efficient. While our algorithm is new the statistical method of estimation we employ is not. In fact, it represents a special case of C. R. Rao's (1971) MINQUE (with V = I). It is also identical (Communication by S. R. Searle) with a special case of the first iterate solution of the REML equations of Corbeil and Searle (1976) whose algorithms appear to involve much larger computational efforts (proportional to n^2). The computational effort is also considerably less than that involved in the M.L. estimation by Hartley and Rao (1967) which is still fairly laborious in spite of the improvements through the W-transformation by Hemmerle and Hartley (1973).

Inspite of its computational simplicity the estimation procedure has numerous "optimality properties". Apart from being a special case of MINQUE other properties are established in Section 6 and the asymptotic consistency is proved in the Appendix under fairly general conditions. The consistency of our estimator makes it convenient as a starting point for a single M.L. cycle to obtain asymptotically fully efficient estimates.

Finally we establish simple conditions for the estimability of all variance components by our method (see Section 6). In this context we observe that with other methods (such as the Henderson 3 method (Henderson (1953)) or the Abbreviated Doolittle and square root method (see e.g. Gaylor, Lucas and Anderson (1970)) estimability depends on the subjective ordering of the components (such as with the Forward Doolittle procedure) and if the ordering is unfortunate the method may fail to yield estimates for certain components while with a different ordering (not attempted) all components may well be estimable.

2. The Mixed ANOVA Model

Employing the currently used notation we write the mixed ANOVA model in the form

$$y = X\alpha + \sum_{i=1}^{c+1} U_i b_i \qquad (1)$$

where y is an n x 1 vector of observations,

X is an n x k matrix of known coefficients,

α is a k x 1 vector of unknown constants,

U_i is an n x m_i matrix of 0, 1 coefficients, and

b_i is an m_i x 1 vector of normal variables from $N(0, \sigma_i^2)$.

Specifically $U_{c+1} = I_n$ and b_{c+1} is an n-vector of "error variables". Moreover the design matrices U_i have precisely one value of 1 in each of their rows and all other coefficients 0. We denote by $m = \sum_{i=1}^{c} m_i$ the total number of random levels.

We may assume without loss of generality that

$$X'X = I \qquad (2)$$

for if (2) is not satisfied we may orthogonalize X by a Gram Schmidt orthogonalization process with a consequential reparameterization of α omitting any linearly dependent columns in the Gram Schmidt process. Usually the first column of X is the column vector with all elements = $1/\sqrt{n}$. It is the objective of the method to compute estimates of the variance components σ_i^2 and the vector α.

3. The Present Method

The essence of the present method is to

(a) Select c + 1 quadratic forms $Q_j(y)$ in the elements of y.

(b) Use the method of synthesis (Hartley (1967), Rao (1968)) to obtain the coefficients k_{ji} in the formulas for $E(Q_j)$ in the form

$$E(Q_j) = \sum_{i=1}^{c+1} k_{ji}\sigma_i^2. \qquad (3)$$

(c) Estimate σ_i^2 by equating the computed Q_j to their expectations i.e. by inverting the system (3) to compute the vector $\hat{\sigma}^2$ with elements $\hat{\sigma}_i^2$

$$\hat{\sigma}^2 = K^{-}Q(y) \qquad (4)$$

from the vector $Q(y)$ with elements $Q_j(y)$ where $K = (k_{ji})$ with rank to be discussed in Section 6 and 7.

(d) Replace any negative elements of $\hat{\sigma}^2$ by 0, with consequences to be discussed in Section 7.

We now give more details for (a), (b) and (c).

(a) The $Q_j(y)$ will be based on contrasts which do not depend on any elements of α. Accordingly we orthogonalize all U_i matrices on X and construct matrices V_i orthogonal on X as follows: Denote by u(t,i) the tth column vector of U_i and by x(r) the rth column vector of X then the columns v(t,i) of V_i are given by

$$v(t,i) = u(t,i) - \sum_{r=1}^{k} x(r)\{x'(r)u(t,i)\}$$

or

$$V_i = U_i - XX'U_i. \qquad (5)$$

We now choose the c + 1 quadratic forms $Q_j(y)$ as

$$Q_j(y) = y'V_jV_j'y = (V_j'y)'V_j'y \qquad (6)$$

$$j = 1, \ldots, c + 1.$$

(b) It follows from the method of synthesis (see Hartley (1967), J.N.K Rao (1968)) that

$$EQ_j(y) = \sum_{i=1}^{c+1} k_{ji}\sigma_i^2 \quad \text{with} \tag{7}$$

$$k_{ji} = \sum_t (V_j'u(t,i))'(V_j'u(i))$$

Now since $v(\tau,j)$ is orthog al on any $x(\rho)$ (i.e. since $v'(\tau,j)x(\rho) =$ we can write the k_{ji} in the alternative orm

$$k_{ji} = \sum_t (V_j'v(t,i))'(V_j'v(i)) \tag{8}$$

$$= \sum_{t\tau}\sum \{v'(\tau,j)v(t,i)\}$$

showing that $k_{ij} = k_{ji}$.

An alternative form of k_{ji} s

$$k_{ji} = \mathrm{tr}\{(V_jV_j')(V_jV_j')\}. \tag{9}$$

We shall show in Section 6 at the symmetrical matrix $K = (k_{ji})$ wi have full rank $c + 1$ if the $n \times n$ matrice V_iV_i' are not linearly dependent.

(c) We shall also show in Sect 6 that the system of equations

$$\underset{\sim}{Q} = K\underset{\sim}{\hat{\sigma}^2} \tag{10}$$

is consistent even if the k of K is degenerate. Solving (10) the form

$$\underset{\sim}{\hat{\sigma}^2} = K^-\underset{\sim}{Q} \tag{11}$$

we shall, of course, be pa cularly interested in the full rank cas when $K^- = K^{-1}$.

4. The Computational L

It may be helpful to give an id of the computational efficiency of the present m od by tabulating the number of products involved in t main operations of the algorithm. To this end we fi note simplified versions for the $k_{c+1,i}$: Obser g that $U_{c+1} = I$ we have from (5) that $V_{c+1} = I - XX'$ d since $X'X = I$ we find that $V_{c+1}V_{c+1}' = I - XX'$ and f ally from (9) that

$$k_{c+1,c+1} = \mathrm{tr}(I - XX')(I - XX') = \mathrm{tr}(I - XX') = n - k. \tag{12}$$

Similarly we find that

$$k_{c+1,i} = \mathrm{tr}\{(I - XX')(V_iV_i')\} = \mathrm{tr}\{V_iV_i' - XX'V_iV_i'\} = \mathrm{tr}V_iV_i'. \tag{13}$$

Further we note the form of $V_{c+1}'y$ i.e.

$$V_{c+1}'y = y - XX'y. \tag{14}$$

Defining now the adjoined matrices

$$U = (U_1 \mid \ldots \mid U_c) \qquad V = (V_1 \mid \ldots \mid V_c) \tag{15}$$

the bulk of the work consists of the formation of the elements of the symmetrical matrix $V'V = V'U = U'V$. The elements of this matrix are assembled in submatrices in accordance with the partition (15) as shown in the Schedule 1 below where it must be remembered that the range of the column index t depends on i and is $t = 1, \ldots, m_i$ and the range of $\tau = 1, \ldots, m_j$ so that the submatrix $V_j'U_i$ has dimensions $m_j \times m_i$. The k_{ji} for $i \geq j = 1, \ldots, c$ are then obtained by forming the sums of squares of the elements in each submatrix in accordance with (7).

Finally, we recite the formulas for the remaining coefficients in the equation (10). The $k_{c+1,c+1}$ and $k_{c+1,i}$ are computed from (12) and (13) respectively and the right hand sides of $Q_j(y)$ from the second form in (6) for $j = 1, \ldots, c$ while $Q_{c+1}(y)$ is given in accordance with (14) by

$$Q_{c+1}(y) = y'y - (X'y)'(X'y). \tag{16}$$

Schedule 1: Submatrices of V'U

	U_1	U_2	...	U_c
V_1	$v(\tau,1)'u(t,1)$	$v(\tau,1)'u(t,2)$	...	$v(\tau,1)'u(t,c)$
V_2		$v(\tau,2)'u(t,2)$	...	$v(\tau,2)'u(t,c)$
.				.....
.				
V_c				$v(\tau,c)'u(t,c)$

We can now summarize the approximate number of products involved in the various operations of the algorithms.

We list the algorithms and show the associated numbers of products in ( ).

1. Orthogonalization of X'X ($k^+(k^+ - 1)n$, where k^+ denotes the number of columns in the original matrix X)
2. Computation of $X'U_i$ for i = 1, ..., c, (0, subtotals of X)
3. Computation of $X(X'U_i)$ for i = 1, ..., c from equation (5), (nmk)
4. Computation of U'V = V'V in accordance with Schedule 1, (0 products since the elements are subtotals of the elements v(t,i))
5. Computation of k_{ij} for i,j = 1, ..., c from equation (7), (1/2(m)(m + 1))
6. Computation of $k_{c+1,i}$ for i = 1, ..., c from equation (13), (mn)
7. Computation of $k_{c+1,c+1}$ from equation (12), (0 products)
8. Computation of the $Q_j(y)$ for j = 1, ..., c+1 from 2nd form of equation (6) and equation (16), ((m + k + 1)(n + 1)).

The important point is that the number of products is only a linear function of the number of data lines n. An approximate formula for the total number of products is $n\{k^+(k^+ - 1) + (m + 1)(k + 1)\}$.

5. A Numerical Example

A small numerical example with n = 4, k^+ = 3, k = 2, c = 1, m_1 = 2, m = 2, m_2 = n = 4 is shown in schedule 2 below.

Schedule 2: A Numerical Example of a Mixed Model

y	X original	U_1	U_2
4	1 1 0	1 0	1 0 0 0
2	1 1 0	0 1	0 1 0 0
1	1 0 1	0 1	0 0 1 0
2	1 0 1	0 1	0 0 0 1

X new		V_1	
(1/2)	(1/2)	+(1/2)	-(1/2)
(1/2)	(1/2)	-(1/2)	+(1/2)
(1/2)	-(1/2)	0	0
(1/2)	-(1/2)	0	0

The orthogonalization of X (original) and X (new) follows the standard Gram Schmidt procedure and reduces the $k^+ = 3$ dependent columns to k = 2 columns which are orthogonal and standardized. Note that

$x(2)_{new} = x(2)_{old} - (1/2)x(1)_{old}$ and

$x(3)_{old} = x(1)_{new} - x(2)_{new}$ must be eliminated.

Using now $x(r) = x(r)_{new}$ we orthogonalize U_1 on X and compute (see (5))

$$x'(1)u(1,1) = +(1/2); \quad x'(2)u(1,1) = +(1/2)$$

and hence

$$v(1,1) = u(1,1) - (1/2)x(1) - (1/2)x(2)$$

likewise

$$x'(1)u(2,1) = (3/2); \quad x'(2)u(2,1) = -(1/2)$$

and hence

$$v(1/2) = u(2,1) - (3/2)x(1) + (1/2)x(2).$$

This yields the matrix V_1 in schedule 2 which has only one independent column. The elements of $V_1'U_1$ require the computation of

$$v(1,1)'u(1,1) = (1/2);$$

$$v(1,1)'u(2,1) = v(2,1)'u(1,1) = -(1/2)$$

and

$$v(2,1)'u(2,1) = (1/2)$$

with sum of squares of $k_{11} = 4(1/2)^2 = 1$. Further (equation (12)) $k_{22} = 4 - 2 = 2$ and (equation (13)) $k_{12} = k_{21} = 4(1/2)^2 + 4(0)^2 = 1$ so that the K matrix is given by $K = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$. Finally, (equation (16))

$$Q_2(y) = 4^2 + 2^2 + 1^2 + 2^2 - ((1/2)9)^2 - ((1/2)3)^2$$

$$= 25 - (90/4) = 25 - 22.5 = 2.5$$

and (equation (6))

$$Q_1(y) = ((1/2)2)^2 + ((1/2)(-2))^2 = 2.$$

The solution of $\underset{\sim}{Q} = K\underset{\sim}{\hat{\sigma}}^2$ therefore yields $\hat{\sigma}_2^2 = 1/2$, $\hat{\sigma}_1^2 = 1.5$.

6. Optimality Properties and the Consistency of the Equations

The estimators described in Section 3 may be seen to be "best at $\sigma_i^2 = 0$, i = 1, ..., c, $\sigma_{c+1}^2 = 1$" as defined by L. R. LaMotte (1973). Therefore, the

consistency of equation (10), regardless of the rank of K, is established as Lemma 4 by LaMotte (1973). That the estimators defined by (11) are "best" among invariant quadratic unbiased estimators guarantees that they are admissible in that class; that is, no other invariant quadratic unbiased estimators have uniformly less variance for all $\underline{\sigma}$. Further, as noted by LaMotte (1973), the estimators (11) have the property that in any model for which a uniformly best estimator exists, (11) will be uniformly best. Finally, it may be seen that the "synthesis" estimators (11) are also MINQUE as in Rao (1971, Section 6) with V = I. No claim is made that this choice of the norm has any particular merits among the rather general family of the norms covered by MINQUE formulas. However, it appears to be reasonable to us that in the absence of any theoretical criteria for selection of MINQUE norms a norm leading to simple estimators may be regarded as meritorious.

Following Section A5 in LaMotte (1973), it may be seen that the rank of K is equal to the number of linearly independent matrices among $V_i V_i'$, $i = 1, \ldots, c + 1$. Thus a singular K may occur if the $U_i U_i'$ matrices are not all linearly independent or if there exists (see (5)) a linear combination of the $U_i U_i'$ matrices whose columns are contained in the linear subspace spanned by the columns of X. In the first case the singularity is caused by the design leading to the U_i matrices, while in the second the singularity is caused by confounding fixed and random effects. In either case, (10) is consistent but some linear combinations of the variance components can not then be unbiasedly estimated. We should stress however that other special cases of MINQUE (not necessarily invariant to α) may also deserve particular attention.

APPENDIX

The Asymptotic Consistency of $\hat{\sigma}^2$

In discussing the asymptotic behavior of $\hat{\sigma}^2$ it is of course necessary to specify the limiting process under which such properties are supposed to hold. Clearly it is necessary for the consistent estimation of the variances $\sigma_i^2 = \text{Var } b_i$ that the number of elements m_i in the vectors b_i all tend to ∞. For the identity matrix U_{c+1} we have $m_{c+1} = n$ the overall sample size. For the remaining m_i we assume that their limiting behavior is related to n by

$$Ln^{1-\alpha_i} \leq m_i \leq Un^{1-\alpha_i} \tag{17}$$

where $0 \leq \alpha_i < 1$ and L,U are universal constants. More specifically we assume that $\alpha_{c+1} = 0$ but $\alpha_i > 0$ for $i = 1, \ldots, c$. Generalizations to situations in which $\alpha_i = 0$ for several components are under consideration. Denote now by

$$\nu(t, i) = \text{number of elements in } u(t, i) \text{ which are 1} \tag{18}$$

and

$$\nu(t, i; \tau, j) = \text{number of rows in which both } u(t, i) \text{ and } u(\tau, j) \text{ have elements 1.} \tag{19}$$

Using these concepts we introduce the following conditions of 'pseudo orthogonality' of the u(t, i) vectors. We assume that

$$\ell n^{\alpha_i} \leq \nu(t, i) \leq u\, n^{\alpha_i} \tag{20}$$

(where ℓ, u are universal constants) and that

$$\nu(t, i; \tau, j) = o(\nu(t, j)) \quad i \neq j \text{ with } i = 1, \ldots, c + 1 \text{ and } j = 1, \ldots, c \tag{21}$$

The relationship between (17) and (20) is obvious since $\sum_{t=1}^{m_i} \nu(t, i) = n$ so that (20) implies (17) with $U = \frac{1}{\ell}$ and $L = \frac{1}{u}$ and the stronger condition (20) implies a uniform order of magnitude for all $\nu(t, i)$ in a given U_i. Since the columns of the U_i matrices

are orthogonal we have $\nu(t, i; \tau, i) = 0$ for all pairs $t \neq \tau$. For columns $u(t, i)$, $u(\tau, j)$ with $i \neq j$ condition (21) is satisfied if there is an asymptotically uniform distribution of the $\nu(t, i)$ rows for which $u(t, i)$ has elements 1 over a fraction qm_j of the m_j columns of U_j where $0 < q < 1$ since the fraction of $\nu(t, i)$ which gives rise to $\nu(t, i; \tau, j)$ will be $O(q^{-1}m_j^{-1}) = O(n^{\alpha_j - 1})$ and will tend to zero.

Next we must introduce conditions on the orthogonal standardized matrix X with elements x_{sr}. Denote by $\sum_{s(t,i)} x_{sr}^2$ the sum of x_{sr}^2 over those rows for which $u(t, i)$ has a 1 element then we assume that

$$\sum_{s(t,i)} x_{sr}^2 = O(n^{\alpha_i - 1}) . \tag{23}$$

Since $\sum_s x_{sr}^2 = 1$ and the number of terms in $\sum_{s(t,i)}$ is $\nu(t, i) = O(n^{\alpha_i})$ condition (23) implies that asymptotically the x_{sr}^2 have a uniform density $x_{sr}^2 = O(n^{-1})$.

Finally we place on record a consequence of conditions (18) to (23): it follows from (5) using (18), (19), (23) and Schwartz's inequality that

$$u'(t, i)\, v(\tau, j) = \begin{cases} \nu(t, i) + O(n^{2\alpha_i - 1}) & \text{for } t = \tau,\ i = j \\ 0 + O(n^{2\alpha_i - 1}) & \text{for } t \neq \tau,\ i = j \\ \nu(t, i; \tau, j) + O(n^{\alpha_i + \alpha_j - 1}) & \text{for } i \neq j \end{cases} \tag{24}$$

We now turn to the asymptotic behavior of the k_{ii} and k_{ij}. From (8), (17), (20), and (25) we have that

$$\begin{aligned} k_{ii} &= \sum_{t=1}^{m_i} \sum_{\tau=1}^{m_j} (u'(t, i)\, v(\tau, i))^2 \\ &= \sum_{t=1}^{m_i} \{u'(t, i)\, v(t, i)\}^2 \\ &\quad + \sum_{t \neq \tau}^{m_i} \{u'(t, i)\, v(\tau, i)\}^2 \\ &\geq \text{Const } n^{1-\alpha_i+2\alpha_i} + O(n^{2-2\alpha_i+4\alpha_i-2}) \\ &\geq C\, n^{1+\alpha_i} \qquad \text{for all } i = 1, \ldots, c+1. \end{aligned} \tag{25}$$

From (8), (17), (19), (21) and (24) we have for $i \neq j$; $i = 1, \ldots, c+1$; $j = 1, \ldots, c$

$$\begin{aligned} k_{ij} &= \sum_{t=1}^{m_i} \sum_{\tau=1}^{m_j} \{u'(t, i)\, v(\tau, j)\}^2 \\ &= \sum_t^{m_i} \sum_\tau^{m_j} \nu(t, i; \tau, j)^2 \\ &\quad + O(n^{\alpha_i+\alpha_j-1}) \sum_t^{m_i} \sum_\tau^{m_j} \nu(t, i; \tau, j) \\ &\quad + \sum_t^{m_i} \sum_\tau^{m_j} O(n^{2\alpha_i+2\alpha_j-2}) \\ &= \sum_t^{m_i} o(\nu(t, i)) \sum_\tau^{m_j} \nu(t, i; \tau, j) \\ &\quad + O(n^{\alpha_i+\alpha_j-1})n + O(n^{2-\alpha_i-\alpha_j})\, O(n^{2\alpha_i+2\alpha_j-2}) \\ &= o(n^{1+\alpha_i}) + O(n^{\alpha_i+\alpha_j}) = o(n^{1+\alpha_i}) \end{aligned} \tag{26}$$

since $\alpha_j < 1$. Similarly we prove by symmetry that $k_{ij} = o(n^{1+\alpha_j})$ for $i \neq j \leq c$. From (25) and (26) it is clear that for all large n the $c \times c$ matrix k_{ij} for $i, j = 1, \ldots, c$ is asymptotically diagonal with diagonal coefficients $\geq cn^{1+\alpha_i}$ while the coefficients $k_{c+1,j}$ are asymptotically equal to $o(n)$. Moreover it is obvious from (12) that $k_{c+1,c+1} \geq Cn$. Using therefore the first c equations of $K\hat{\sigma}^2 = Q(y)$ we obtain that

$$\begin{aligned} \hat{\sigma}_i^2 &= O(n^{-\alpha_i - 1})\{Q_i(y) - o(n)\hat{\sigma}_{c+1}^2\} \\ &= O(n^{-\alpha_i - 1})Q_i(y) + o(n^{-\alpha_i})\hat{\sigma}_{c+1}^2 \end{aligned} \tag{27}$$

$$\text{for } i = 1, \ldots, c .$$

Substituting (27) in the last equation we obtain

$$\hat{\sigma}_{c+1}^2\{cn + o(n^{1-\alpha_{\min}})\} = Q_{c+1}(y) + \sum_{i=1}^{c} Q_i(y)\, o(n^{-\alpha_i}) \tag{28}$$

or

$$\hat{\sigma}_{c+1}^2 = O(n^{-1})Q_{c+1}(y) + \sum_{i=1}^{c} Q_i(y)\, o(n^{-\alpha_i - 1}). \tag{29}$$

Substituting (29) back in (27) we obtain

$$\hat{\sigma}_i^2 = O(n^{-\alpha_i - 1})Q_i(y) + o(n^{-1-\alpha_i})Q_{c+1}(y) . \tag{30}$$

Equations (29) and (30) show that $\hat{\sigma}^2$ is estimable from the $Q_i(y)$. They also show that $\hat{\sigma}^2$ is consistent provided we can show that

$$\text{Var } Q_r(y) = o(n^{2\alpha_r+2}) \quad \text{for } r = 1, \ldots, c \qquad (31)$$

$$\text{Var } Q_{c+1}(y) = o(n^2)$$

since Cov $Q_i(y)Q_j(y) = 0(\text{Var } Q_i(u)^{\frac{1}{2}} \text{ Var } Q_j(y)^{\frac{1}{2}})$.

In order to prove the first result in (31) we use formulas [22], [32], [33] and [34] of J.N.K. Rao (1968) with slightly altered notation. Formula [22] gives $E\, Q_r^2(y)$ in the form

$$E(Q_r(y)^2) = 2 \sum_{i<j=1}^{c+1} \sum c_{ij}\sigma_i^2\sigma_j^2 + \sum_{i=1}^{c+1} c_{ii}\sigma_i^4 + \sum_{i=1}^{c+1} h_i\mu_{4i} \qquad (32)$$

where $\mu_{4i} = E\, b_{i\ell}^4$ are the 4^{th} moments of the elements $b_{i\ell}$ of b_i. Noting that $\text{Var } Q_r(y)^2 = E\, Q_r(y)^2 - E^2(Q_r(y))$ the leading terms of c_{ii} and c_{ij} given by J.N.K. Rao's equations [33] and [32] cancel and we are left to consider the orders of magnitude of

$$\dot{c}_{ii} - 2h_i = \sum_{t<\tau=1}^{m_i} \sum \{Q_r(u(t, i) + u(\tau, i)) - Q_r(u(t, i)) - Q_r(u(\tau, i)\}^2$$

$$= \sum_{t<\tau=1}^{m_i} \sum \{ \sum_{s=1}^{m_r} 2(u(t, i)' v(s, r)) \times (u(\tau, i)' v(s, r))\}^2 . \qquad (33)$$

Consider first the case $r = i$. We distinguish two terms when $s = t$ and $s = \tau$. For those two terms $(u(t, i)' v(s, i)); (u(\tau, i)' v(s, i))$ is from (24) of the order of magnitude $0(n^{\alpha_i}) 0(n^{2\alpha_i-1}) = 0(n^{3\alpha_i-1})$. For the remaining terms in $\sum_{s=1}^{m_r}$ the product is of the order $0(n^{4\alpha_i-2})$ but the number of terms is of the order $0(n^{1-\alpha_i})$ so that $\{\Sigma\}^2$ is $0(n^{6\alpha_i-2})$ and hence $\dot{c}_{ii} = 0(n^{2-2\alpha_i}) 0(n^{6\alpha_i-2}) = 0(n^{4\alpha_i}) = o(n^{2\alpha_i+2})$ since $\alpha_i < 1$.

Consider next the case $r \neq i$ and $r \neq c + 1$. We have from (33) and (24)

$$\dot{c}_{ii} = \sum_{t<\tau}^{m_i} \sum \{\sum_s (\nu(t, i; s, r)\, \nu(t, i; s, r) + 0(n^{2\alpha_i+2\alpha_r-2}) + 0(n^{\alpha_i+\alpha_r-1})(\nu(t, i; s, t) + \nu(\tau, i; s, r))\}^2$$

$$= \sum_{t<\tau}^{m_i} \sum \{o(\nu(s, r)) \sum_s \nu(\tau, i; s, r) + 0(n^{2\alpha_i+\alpha_r-1}) + 0(n^{\alpha_i+\alpha_r-1})(\nu(t, i) + \nu(\tau, i))\}^2$$

$$= \sum_{t<\tau}^{m_i} \sum \{o(n^{\alpha_r+\alpha_i}) + 0(n^{2\alpha_i+\alpha_r-1})\}^2 \qquad (34)$$

$$= o(n^{2+2\alpha_r}) + o(n^{\alpha_i+2\alpha_r+1}) + 0(n^{2\alpha_i+2\alpha_r})$$

$$= o(n^{2+2\alpha_r}) .$$

The case $r \neq i$, $r = c + 1$ follows on the same lines as (34) except that $\alpha_r = 0$ and that $\nu(t, i; s, c + 1)\, \nu(\tau, i; s, c + 1) = 0$ since $u(s, r)$ has a 1 only in the s^{th} row and either $u(t, i)$ or $u(\tau, i)$ have a zero in that row. The order of magnitude of $\{\}$ will therefore be $0(n^{2\alpha_i-1})$ and $\dot{c}_{ii}$ will be $0(n^{2\alpha_i}) = o(n^2)$.

The treatment of the c_{ij} in J.N.K. Rao's formula [33] follows on similar lines to the above proof for the c_{ii} if of the two alternatives $i < j$, $j < i$ in (21) the smaller α_i, α_j is selected for majorisations. It remains to consider the terms

$$h_i = \sum_{t=1}^{m_i} Q_r^2(u(t, i)) = \sum_{t=1}^{m_i} \{ \sum_{s=1}^{m_r} (u'(t, i)\, v(s, r))^2\}^2 . \qquad (35)$$

For the case $r = i$ we have using (24)

$$h_i = \sum_{t=1}^{m_i} \{(u'(t, i)\, v(t, i))^2 + \sum_{s\neq t}^{m_i} (u'(t, i)\, v(s, i))^2\}^2 \qquad (36)$$

$$= \sum_{t=1}^{m_i} \{0(n^{2\alpha_i}) + 0(n^{3\alpha_i-1})\}^2$$

$$= 0(n^{1+3\alpha_i}) + 0(n^{4\alpha_i}) + 0(n^{5\alpha_i-1})$$

$$= o(n^{2\alpha_i+2}) = o(n^{2\alpha_r+2}) \quad \text{for } i = r \neq c+1,$$

$$= o(n^2)$$

For the case $i \neq r$ and $r \neq c+1$

$$h_i = \sum_{t=1}^{m_i} \{ \sum_{s=1}^{m_r} (\nu(t, i; s, r) + O(n^{\alpha_i+\alpha_r-1}))^2\}^2$$

$$= \sum_{t=1}^{m_i} \{ \sum_{s=1}^{m_r} o(\nu(s, r))\nu(t, i; s, r) + O(n^{\alpha_i+\alpha_r-1}) \sum_s \nu(t, i; s, r) + O(n^{1-\alpha_r}) O(n^{2\alpha_i+2_r-2})\}^2 \quad (37)$$

$$= \sum_{t=1}^{m_i} \{o(n^{\alpha_i+\alpha_r}) + O(n^{2\alpha_i+\alpha_r-1})\}^2$$

$$= o(n^{\alpha_i+2\alpha_r+1}) + o(n^{2\alpha_i+2\alpha_r}) + O(n^{3\alpha_i+2\alpha_r-1})$$

$$= o(n^{2+2\alpha_r}).$$

Finally for $r = c+1$, $i \neq r$ we have

$$h_i = \sum_{t=1}^{m_i} \{ \sum_{s=1}^{n} (\nu(t, i; s, r) + O(n^{\alpha_i-1}))^2\}^2 \quad (38)$$

$$= \sum_{t=1}^{m_i} \{ \sum_s \nu(t, i; s, r)^2 + \sum_s \nu(t, i; s, r)\, O(n^{\alpha_i-1}) + O(n^{2\alpha_i-1})\}^2$$

Now since $\nu(t, i; s, c+1)$ is either 0 or 1 we have that $\sum_s \nu(t, i; s, c+1)^2 = \sum_s \nu(t, i; s, c+1)$ $= \nu(t, i)$ so that

$$h_i = \sum_{t=1}^{m_i} \{O(n^{\alpha_i}) + O(n^{2\alpha_i-1})\}^2 \quad (39)$$

$$= O(n^{1-\alpha_i})\, O(n^{2\alpha_i})$$

$$= o(n^2).$$

Since $\hat{\sigma}^2$ is unbiassed and Cov $(\hat{\sigma}^2) \to 0$ as $n \to \infty$ it follows that $\hat{\sigma}^2$ is consistent. Moreover if we replace any negative $\hat{\sigma}_i^2$ by 0 the resulting statistic say $\tilde{\sigma}_i^2$ has a smaller mean square error and hence is also consistent.

The consistent estimator $\tilde{\sigma}^2$ may serve as a starting value for the iterative maximum likelihood estimation procedure described by Hemmerle and Hartley (1973). Under certain regularity conditions (not discussed here) one single cycle of the iteration will result in asymptotically efficient estimators of σ^2 and α. If the iteration is carried to convergence solutions of the ML equations are reached. If no ML cycles are performed a consistent estimator $\tilde{\alpha}$ of α can be computed from the generalized least squares (ML) equations.

$$\tilde{\alpha} = (X'H^{-1}X)^{-1} (X'H^{-1}y) \quad (40)$$

$$\text{where } H = I_n + \sum_{i=1}^{c} \frac{\tilde{\sigma}_i^2}{\tilde{\sigma}_{c+1}^2} U_i U_i'$$

It has been shown by Hemmerle and Hartley (1973) that (40) can be computed directly from the $U_i U_i'$ and $X'U_i$ matrices without the inversion of the $n \times n$ matrix H using their so called W transformation. In fact the W_o matrix (their equation (19)) is essentially given by the $V_i'V_i$ matrices (see the above Schedule 1) and by the contrasts $V_i'y$ required in the computation of $Q_i(y)$.

The variance covariance matrix of $\tilde{\alpha}$ can likewise be computed through the W transformation.

Acknowledgement

One of us (H.O.H.) wishes to acknowledge support from the Army Research Office.

J.N.K. Rao wishes to acknowledge support from the National Research Council of Canada.

References

Corbeil, R. R. and Searle, S. (1976). Restricted maximum likelihood (REML) estimation of variance components in the mixed model. Technometrics 15, 819-826.

Gaylor, D. W., Lucas, H. L. and Anderson, R. L. (1970). Calculation of the expected mean squares by the abbreviated Doolittle and square root methods. Biometrics, 26, 641-656.

Hartley, H. O. (1967). Expectations variances and covariances of ANOVA mean squares by 'synthesis'. Biometrics, 23, 105-114.

Hartley, H. O. (1977). Analysis of unbalanced experiments. Invited address 23rd Conference on the Design of Experiments in Army Research Development and Testing to be published in proceedings.

Hartley, H. O. and Rao, J. N. K. (1967). Maximum likelihood estimation for the mixed analysis of variance model. Biometrika, 54, 93-108.

Hartley, H. O. and Rao, J. N. K. (1977). The estimation of non-sampling variance components in sample surveys. Presented at Symposium on Sample Surveys and Measurement, Chappel Hill, April 14-17, 1977.

Hemmerle, W. J. and Hartley, H. O. (1973). Computing maximum likelihood estimates for the mixed AOV model using the W-transformation. Technometrics, 15, 819-831.

Henderson, C. R. (1953). Estimation of variance and covariance components. Biometrics, 9, 226-252.

LaMotte, L. R. (1973). Quadratic estimation of variance components. Biometrics, 29, 311-330.

Liu, L. and Senturia, J. (1976). Computation of minque variance components estimates. Technical Report #408, Department of Statistics, University of Wisconsin.

Rao, C. R. (1971). Estimation of variance and covariance components-minque theory. Journal of Multivariate Analysis, 1, 257-275.

Rao, J. N. K. (1968). On expectations variances and covariances of ANOVA mean squares by 'synthesis'. Biometrics, 24, 963-978.

MIXED MODEL ALGORITHMS FOR ESTIMATING NON-HOMOGENEOUS VARIANCES

William J. Hemmerle and Brian W. Downs
University of Rhode Island

Abstract

The mixed analysis of variance model with non-homogeneous error variances is studied with respect to maximum likelihood and restricted maximum likelihood estimation. Algorithms which obtain estimators of the variance components and error variances are described.

1. INTRODUCTION

This paper considers the problem of obtaining estimates of the parameters in the general mixed analysis of variance model when the variances associated with the random errors are unequal between groups of observations. Attention is focused upon maximum likelihood (ML) and restricted maximum likelihood (REML) estimators. We initially relate the computations for the case of non-homogeneous variances to that of equal variances through introducing a dummy parameter into the computations. General implementation of ML or REML mixed model algorithms is then considered briefly in section 4. The remainder of the paper deals with special manifestations of the non-homogeneous case--measuring instruments models in particular. It summarizes algorithms which have been developed for these special cases which are much more efficient than the general approach given initially.

2. THE MIXED A.O.V. MODEL WITH UNEQUAL ERROR VARIANCES

Following Hartley and Rao [6] and subsequent authors we write the general mixed A.O.V. model as

$$(2.1)\quad y = X\alpha + U_1 b_1 + U_2 b_2 + \cdots + U_c b_c + e$$

where y is an n vector of observations;

X is an nxp matrix of known fixed numbers, $p \leq n$;

U_i is an nxm_i matrix of known fixed numbers, $m_i \leq n$;

α is a p vector of unknown constants;

b_i is an m_i vector of independent variables from $N(0,\sigma_i^2)$.

In the equal variance case, we assume that e is an $nx1$ vector from $N(0,\sigma^2)$; however, we will generalize to the case when these error variances are unequal between groups of data and partition the model (2.1) such that

$$(2.2)\quad y=\begin{pmatrix} y_1 \\ \hline y_2 \\ \hline \vdots \\ \hline y_k \end{pmatrix},\quad X=\begin{pmatrix} X_1 \\ \hline X_2 \\ \hline \vdots \\ \hline X_k \end{pmatrix},\quad U_i=\begin{pmatrix} U_{i1} \\ \hline U_{i2} \\ \hline \vdots \\ \hline U_{ik} \end{pmatrix}_k,\quad e=\begin{pmatrix} e_1 \\ \hline e_2 \\ \hline \vdots \\ \hline e_k \end{pmatrix}.$$

where y_i is an n_i vector with $\sum_{i=1}^{k} n_i = n$.

We then assume that e_i has the multivariate normal distribution

$$(2.3)\quad e_i \sim MVN(0, s_i^2 I),\ i = 1, \cdots, k$$

and that the random vectors $b_1, b_2, \cdots, b_c, e_1, e_2, \cdots, e_k$ are mutually independent. We let

$$(2.4)\quad \sum_{i=1}^{c} m_i = m$$

and also make certain rank assumptions with respect to X and the U_i's. These rank assumptions are discussed in Miller [12] and also in Hemmerle and Downs [9] as they relate to the non-homogeneous variance case.

From the model (2.1) and the assumptions related to it, the n vector y must have the multivariate normal distribution or likelihood function

$$(2.5)\quad L=\frac{1}{(2\pi)^{n/2}|\Sigma|^{\frac{1}{2}}}\exp\{-(y-X\alpha)'\Sigma^{-1}(y-X\alpha)/2\}$$

with variance-covariance matrix Σ given by

$$(2.6)\quad \Sigma=(D^*+\sigma_1^2U_1U_1'+\sigma_2^2U_2U_2'+\cdots+\sigma_c^2U_cU_c')$$

where D^* is the diagonal matrix whose ith diagonal block, D_{ii}^*, is given by

$$(2.7)\quad D_{ii}^* = s_i^2I_{n_i},\quad i = 1, 2, \cdots, k.$$

3. INTRODUCING A DUMMY PARAMETER

The matrix Σ given by (2.6) may be represented as

$$(3.1)\quad \Sigma = (s_1^2E_1E_1' + \cdots + s_k^2E_kE_k' + \sigma_1^2U_1U_1' + \cdots + \sigma_c^2U_cU_c')$$

where E_i is the nxn_i matrix

$$(3.2)\quad E_i = (0|\cdots|0|I_{n_i}|0|\cdots|0)'$$

such that I_{n_i} is the ith sub-matrix and

$$(3.3)\quad (E_1|E_2|\cdots|E_k) = I_n.$$

We now introduce a "dummy" parameter σ^2 and write (3.1) as

$$(3.4)\quad \Sigma=[\sigma^2I+(s_1^2-\sigma^2)E_1E_1'+\cdots+(s_k^2-\sigma^2)E_kE_k' +\sigma_1^2U_1U_1'+\cdots+\sigma_c^2U_cU_c']$$

If we then let

$$(3.5)\quad \delta_i = (s_i^2-\sigma^2)/\sigma^2,\quad i = 1,2,\cdots,k;$$

$$(3.6)\quad \gamma_i = \sigma_i^2/\sigma^2,\quad i = 1,2,\cdots,c;$$

and factor out σ^2 in (3.4) we obtain

$$(3.7)\quad \Sigma = \sigma^2H;$$

where

$$(3.8)\quad H = [I+\delta_1E_1E_1'+\cdots+\delta_kE_kE_k'+\cdots +\gamma_1U_1U_1'+\cdots+\gamma_cU_cU_c'].$$

The structure of (3.7) and (3.8) in terms of the new variables σ^2, δ_1, $\cdots$, δ_k, γ_1, $\cdots$, γ_c now conforms to the structure of the variance-covariance matrix for the general mixed model with equal variances. The maximum likelihood estimators of these new variables are then constrained such that

$$(3.9)\quad \hat{\sigma}^2 \geq 0,$$

$$(3.10)\quad \hat{\delta}_i \geq 0,\quad i = 1,2,\cdots,k,$$

$$(3.11)\quad \hat{\gamma}_i \geq 0,\quad i = 1,2,\cdots,c,$$

and we maximize the log-likelihood λ given by

$$(3.12)\quad 2\lambda = -n\ln2\pi - n\ln\sigma^2 - \ln|H| - (y-X\alpha)'H^{-1}(y-X\alpha)/\sigma^2$$

subject to these constraints. Hemmerle and Hartley's W transformation [7] may be applied to obtain the necessary partials or their expected values for the optimization method used.

Clearly, we have overparametrized in introducing the dummy parameter σ^2 and there is an infinite number of solutions that yield the same maximum value for λ given by (3.12); however, we need only find one of these solutions in order to find the corresponding unique solution for the $\hat{s}_i^2$'s and $\hat{\sigma}_i^2$'s via the transformations (3.5) and (3.6).

In the cases we have examined, we have found that different starting values for the maximization algorithm produce different final values for the $\hat{\delta}_i$'s, $\hat{\gamma}_i$'s, and $\hat{\sigma}^2$. The latter, however, yielded the same value for the log-likelihood (3.12) and transformed into the same values for the $\hat{s}_i^2$'s and $\hat{\sigma}_i^2$'s. Although we have had success with this procedure, we do not claim that it is foolproof. If one of the s_i^2's wants to converge to zero, then $\hat{\sigma}^2$ should also approach zero since the constraint (3.10) implies

$$(3.13)\quad \hat{s}_i^2 \geq \hat{\sigma}^2\quad i = 1,2,\cdots,k.$$

The estimates of the variance components, the $\hat{\sigma}_i^2$, may of course converge to zero without $\hat{\sigma}^2$ approaching zero. Several numerical examples using this procedure are given in [9].

In order to obtain restricted maximum likelihood (REML) estimators we would maximize λ_1 where

$$(3.14)\quad 2\lambda_1 = -(n-p)\ln2\pi - (n-p)\ln\sigma^2 - \ln|THT'| - y'T'(THT')^{-1}Ty/\sigma^2$$

and T is the same transformation matrix used by Corbeil and Searle [2]. These authors have shown how to obtain REML estimators for the mixed model through use of the W transformation. With the exception of matrix initialization, the basic computations for ML and REML are equivalent.

4. GENERAL IMPLEMENTATION

We have seen that ML (or REML) estimates for the model (2.1) with unequal group variances may be computed using an algorithm constructed for the equal variance case. In latter sections of this paper, we will consider algorithms for important special mixed models with non-homogeneous group variances which are substantially more efficient than the general treatment. At this point, however, we will highlight some of the studies that have been conducted into the different optimization options available for the general case. The data and computations which

support our conclusions are given in the M.S. thesis of Downs [3].

All of these options utilize the W transformation of Hemmerle and Hartley [7] in computing 1st order partials and 2nd order partials or expected values of 2nd order partials. Jennrich and Sampson [10] have already presented evidence that use of the full set of partials (for the fixed effects and σ^2 as well as the variance components) is apt to be more efficient than the two-stage procedure used in [7] to obtain estimates. We also concluded that a one-stage process, with the full partials, is usually more effective; however, in the non-homogeneous variance case we had fewer convergence problems, related to constraining the dummy variable $\hat{\sigma}^2$ to being non-negative, using the two-stage procedure. This quantity is then always evaluated as a quadratic form which is non-negative.

The initial scoring steps suggested in [7] and implemented in [10] seem to be a very effective combination with ultimate Newton-Raphson steps. Of particular interest to us were different computational options with respect to non-negative constraints for the variance estimators. During this research, we were provided an initial version, BMDQ3V, of Jennrich and Sampson's program BMDP3V [11] which we used as a benchmark. Significantly, the square root transformation suggested in [6] and used in [7] produced disappointing results for several examples which were handled very effectively with Jennrich and Sampson's constraining procedures.

Basically, our studies were a testimonial to the optimization method and constraining procedures used in the BMDP3V program; however, it seems that the efficiency of the latter program could be increased were it to incorporate the Cholesky algorithm for the W transformation developed by Hemmerle and Lorens [8]. We have used this algorithm along with the optimization method and constraining procedures used in BMDP3V and get equivalent results in essentially the same number of iterations but do each iteration faster. Forming the W matrix consumes the lion's share of the computations at each iterative step of maximizing the log-likelihood. The Cholesky algorithm will reduce the number of operations (multiplications/divisions) required to form W by a factor of 4 when m is large with respect to p.

To confirm this increase in efficiency, we conducted a few benchmark runs of the maximization computations obtaining the execution times required solely for the iterative steps; these times did not include the time necessary for input, set-up, computation of ancillary statistics and output. Table 1 summarizes the results and characteristics of the three data sets used. The times represent an average of 8 runs for each data set. We allowed BMDQ3V to terminate and then ran our program for the same number of iterations. Computed estimates agreed to at least 6 digits. Our program performs very well on data set 3 where 3 components are quickly estimated as being near zero since sequential Cholesky steps are not performed for near zero component estimates. Similar savings are apparently not realized by BMDQ3V because of its particular formulation of the W transformation.

We conclude this section with a brief review of the Cholesky W transformation algorithm.

In computing the elements of the $(m+p+1)\times(m+p+1)$ matrix

$$W = \left(\begin{array}{c|c|c} V'H^{-1}V & V'H^{-1}X & V'H^{-1}y \\ \hline X'H^{-1}V & X'H^{-1}X & X'H^{-1}y \\ \hline y'H^{-1}V & y'H^{-1}X & y'H^{-1}y \end{array}\right) \tag{4.1}$$

for given values of the parameters, we first form the matrix

$$W_o = \left(\begin{array}{c|c|c} V'V & V'X & V'y \\ \hline X'V & X'X & X'y \\ \hline y'V & y'X & y'y \end{array}\right) \tag{4.2}$$

where

$$V = [U_1|U_2|\cdots|U_c]. \tag{4.3}$$

Let us partition W_o as

$$W_o = \left(\begin{array}{c|c} A_o & B_o \\ \hline B_o' & C_o \end{array}\right) \tag{4.4}$$

where A_o is the mxm matrix $V'V$. Suppose that the matrix W is similarly partitioned such that

$$W = \left(\begin{array}{c|c} W_{11} & W_{12} \\ \hline W_{12}' & W_{22} \end{array}\right) \tag{4.5}$$

where W_{11} is an mxm matrix. We add D^{-1} to the upper left-hand corner of W_o to form

$$\bar{W}_o = \left(\begin{array}{c|c} D^{-1}+A_o & B_o \\ \hline B_o' & C_o \end{array}\right) \tag{4.6}$$

and then perform m sequential Cholesky steps on (4.6) operating upon all of the m+p+1 rows and columns of this matrix. This results in the decomposition of $\bar{W}_o$ as

$$\bar{W}_o = \left(\begin{array}{c} T_{11}' \\ \hline T_{12}' \end{array}\right)\left(T_{11}|T_{12}\right) + \left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & Z \end{array}\right) \tag{4.7}$$

where T_{11} is upper triangular of order m; T_{11}, T_{12}, and Z would overwrite $(D^{-1}+A_o)$, B_o, and C_o respectively in storage. The following relationships may then be obtained.

$$W_{11} = D^{-1} - (D^{-1}T_{11}^{-1})(D^{-1}T_{11}^{-1})' \tag{4.8}$$

(4.9) $$W_{12} = D^{-1}T_{11}^{-1}T_{12}$$

(4.10) $$W_{22} = Z$$

All of the calculations, including those for (4.8), (4.9), and (4.10), may be done "in place" using only the storage required for the upper (or lower) triangular portion of W_o. This algorithm to form W is of the order $m^3/2$ when m is large with respect to p. It extends in a simple manner to handle non-negative constraints, zero or near zero components.

5. PROPORTIONAL VARIANCES

When the variances are proportional such that

(5.1) $$s_i^2 = \ell_i\sigma^2, \quad i = 1, 2, \cdots, k$$

and the ℓ_i's are known, the variance-covariance matrix for the model (2.1) becomes

(5.2) $$\sigma^2H = \sigma^2(L+VDV')$$

where V is given by (4.3) and D and L are diagonal matrices whose ith diagonal blocks, D_{ii} and L_{ii}, respectively, are given by

(5.3) $$D_{ii} = \gamma_i I_{m_i}, \quad i = 1, 2, \cdots, c,$$

(5.4) $$L_{ii} = \ell_i I_{n_i}, \quad i = 1, 2, \cdots, k,$$

with γ_i defined as in (3.6).

In the equal variance case we form the matrix W_o given by (4.2) initially and then apply the W transformation at each iteration to simply compute the forms

(5.5) $$U_i'H^{-1}U_j,\ U_i'H^{-1}X,\ U_i'H^{-1}y,$$
$$X'H^{-1}X,\ X'H^{-1}y,\ y'H^{-1}y$$

needed in the calculation of first order partials, second order partials, or expected values of second order partials for the optimization step. For the proportional variance case we first form the matrix

(5.6) $$W_o^* = \left(\begin{array}{c|c|c} V'L^{-1}V & V'L^{-1}X & V'L^{-1}y \\ \hline X'L^{-1}V & X'L^{-1}X & X'L^{-1}y \\ \hline y'L^{-1}V & y'L^{-1}X & y'L^{-1}y \end{array}\right)$$

and then proceed exactly as in the equal variance case in applying the W transformation to obtain the forms (5.5) required for the partials. In order to evaluate the log-likelihood in the equal variance case, $|H|$ is evaluated as

(5.7) $$|H| = |D|\cdot|D^{-1}+V'V|$$

where $|D^{-1}+V'V|$ is a by-product of the W transformation. In the proportional variance case we must compute $|H|$ as

(5.8) $$|H| = |L|\cdot|D|\cdot|D^{-1}+V'L^{-1}V|$$

where, in this case, $|D^{-1}+V'L^{-1}V|$ would be a by-product of the W transformation.

In essence what this says is that if we weight the observations, the rows of X, and the rows of indicator variables in V by the reciprocal of the square root of the corresponding ℓ_i before we process the data, we will obtain the proper maximum likelihood estimators for σ^2 and the γ_i's as well as for the α vector. Later in the sequel, we will use the relationships given in this section in developing algorithms for cases when the ℓ_i's are unknown.

6. SPECIAL MODELS: BALANCED DATA

In the remaining sections of this paper we will concentrate on special cases of the model (2.1) with non-homogeneous variances that have received attention in the literature. In particular, we consider those models studied by Grubbs [4], [5] and others for assessing the imprecision of measuring instruments. We will summarize algorithms which have been developed in Hemmerle and Downs [9] and Downs [3], for both balanced and unbalanced data, which capitalize upon the structure of these models to provide more efficient computation than the general algorithm discussed in previous sections.

The model considered by Grubbs [5] in using 3 instruments to make simultaneous measurements on each of a series of N items is the mixed model

(6.1) $$y_{ij} = \alpha_i + \beta_j + e_{ij}$$

where: $i = 1, 2, 3$ and $j = 1, 2, \cdots, N$; α_i is fixed; β_j is random with variance component σ_β^2; and $\mathrm{Var}(e_{ij}) = s_i^2$. We will consider the same model for the more general case when

(6.2) $$i = 1, 2, \cdots, M.$$

Application of the general non-homogeneous variances algorithm described in sections 2 and 3 consists essentially of augmenting the design matrix for the variance components with a partitioned identity matrix. As a consequence, the nxn inverse matrix H^{-1} is in fact computed explicitly even though the W transformation is applied. Since n is the number of observations ($n = M\cdot N$ for the model (6.1)), the computational requirements can become excessive. We intend to exploit the balanced data structure of the model (6.1), with the generalization specified by (6.2), to avoid explicit calculation of H^{-1} for both balanced and unbalanced data. Towards this end, we present some relationships for the model (6.1) which are easily confirmed.

Using the same notation as (5.1) and (5.2), the variance-covariance matrix H may be written as

$$(6.3)\quad H=\begin{pmatrix}(\gamma_1+\ell_1)I_N & \gamma_1 I_N & \cdots & \gamma_1 I_N\\ \gamma_1 I_N & (\gamma_1+\ell_2)I_N & \cdots & \gamma_1 I_N\\ \vdots & \vdots & \ddots & \vdots\\ \gamma_1 I_N & \gamma_1 I_N & \cdots & (\gamma_1+\ell_M)I_N\end{pmatrix}.$$

The inverse of (6.3), H^{-1}, is similarly composed of blocks of NxN diagonal matrices. Let us denote the ijth block of H^{-1} corresponding to the ijth block of H by

$$(6.4)\quad (\text{ijth block of } H^{-1}) = a_{ij}I_N.$$

Then if we let

$$(6.5)\quad u = \prod_{i=1}^{M}\ell_i,\quad v = \sum_{i=1}^{M}(1/\ell_i),$$

the a_{ij}'s in (6.4) are given by

$$(6.6)\quad a_{ij} = \delta_{ij}/\ell_i - \gamma_1/(1+v\gamma_1)\ell_i\ell_j,$$

where δ_{ij} is the Kronecker delta, and the determinant of (6.3) may be obtained as

$$(6.7)\quad |H| = q^N$$

where

$$(6.8)\quad q = u(1+v\gamma_1).$$

For example, when M = 3 the variance-covariance matrix (6.3) has as its inverse

$$(6.9)\quad H^{-1} = \frac{1}{q}\begin{pmatrix}(\gamma_1\ell_2+\gamma_1\ell_3+\ell_2\ell_3)I_N & -\gamma_1\ell_3 I_N & -\gamma_1\ell_2 I_N\\ -\gamma_1\ell_3 I_N & (\gamma_1\ell_1+\gamma_1\ell_3+\ell_1\ell_3)I_N & -\gamma_1\ell_1 I_N\\ -\gamma_1\ell_2 I_N & -\gamma_1\ell_1 I_N & (\gamma_1\ell_1+\gamma_1\ell_2+\ell_1\ell_2)I_N\end{pmatrix};$$

where

$$(6.10)\quad q = \gamma_1\ell_1\ell_2+\gamma_1\ell_1\ell_3+\gamma_1\ell_2\ell_3+\ell_1\ell_2\ell_3$$

Using the solution given by (6.4) through (6.6) for H^{-1}, one can show that the maximum likelihood estimators for the α_i's in (6.1) are the analysis of variance estimators

$$(6.11)\quad \hat{\alpha}_i = \frac{1}{N}\sum_{j=1}^{N} y_{ij},\quad i=1,2,\cdots,M.$$

An additional relationship which we will use in what follows is the explicit determination of W_o^* given by (5.6) for the model (6.1). This matrix is given by

$$(6.12)\quad W_o^* =$$

$$\left(\begin{array}{ccc|ccc|c} \Sigma 1/\ell_1 & \cdots & 0 & 1/\ell_1 & \cdots & 1/\ell_M & \Sigma y_{i1}/\ell_i\\ & \ddots & \vdots & \vdots & & \vdots & \vdots\\ & & \Sigma 1/\ell_i & 1/\ell_1 & \cdots & 1/\ell_M & \Sigma y_{iN}/\ell_i\\ \hline & & & N/\ell_1 & \cdots & 0 & (1/\ell_1)\Sigma y_{1j}\\ & (\text{SYMMETRIC}) & & & \ddots & \vdots & \vdots\\ & & & & & N/\ell_M & (1/\ell_M)\Sigma y_{Mj}\\ \hline & & & & & & \Sigma\Sigma y_{ij}^2/\ell_i \end{array}\right)$$

The concept of the balanced algorithm is very simple. We appeal to the case of proportional variances treated in section 5 and consider the ℓ_i's as fixed in obtaining new approximations for $\hat{\sigma}^2$ and $\hat{\gamma}_1$ (in the unbalanced case to follow we must also obtain new approximations for $\hat{\alpha}$). The matrix W_o^* given by (6.12) is easily formed for a given set of ℓ_i's and the W transformation may then be utilized, using W_o^*, in obtaining the new values for $\hat{\sigma}^2$ and $\hat{\gamma}_1$. In order to obtain new approximations for the ℓ_i's, for use in the next iteration, we initially compute the a_{ij}'s or H^{-1} given by (6.6) and store these M(M+1)/2 coefficients. These coefficients are then utilized in computing the 1st order ℓ_i partials as well as the 2nd order ℓ_i partials (or expected values of these 2nd order partials) for a Newton-Raphson step (or Fisher scoring step) to improve the ℓ_i approximations. Both $\hat{\sigma}^2$ and $\hat{\gamma}_1$ are included in the ℓ_i partials.

Due to the structure of H^{-1} given by (6.4), these ℓ_i partials are easily computed. If we let

$$(6.13)\quad d = (y-X\alpha),$$

for ML estimation we have that:

$$(6.14)\quad \frac{\partial\lambda}{\partial\ell_i} = -\frac{N}{2}a_{ii} + \frac{1}{2\sigma^2}\left\{\sum_{j=1}^{N}\left(\sum_{k=1}^{M}d_{kj}a_{ki}\right)^2\right\}.$$

$$(6.15)\quad \frac{\partial^2\lambda}{\partial\ell_i\partial\ell_j} =$$

$$\frac{N}{2}a_{ij}^2 - \frac{a_{ij}}{\sigma^2}\left(\sum_{k=1}^{N}\sum_{\ell=1}^{M}d_{\ell k}a_{\ell i}\sum_{m=1}^{M}d_{mk}a_{mj}\right)$$

$$(6.16)\quad E\left(\frac{\partial^2\lambda}{\partial\ell_i\partial\ell_j}\right) = -\frac{N}{2}a_{ij}^2$$

where a_{ij} is given by (6.6).

As an example of the balanced ℓ_i algorithm, we use the data given in Grubbs [5], in which he records the measurements taken by three velocity chronographs on each of twelve successive rounds fired from a 155 mm gun. The model is (6.1) with M=3 and N=12. Using his estimation procedure, Grubbs obtains the values .0065, .0525, .2186, and 2.0164 as estimates of s_1^2, s_2^2, s_3^2, and σ_1^2, respectively. A summary of our computations to obtain the maximum likelihood

estimators for these quantities is given in Table 2 for the set of starting values $\hat{\ell}_1=1$, $\hat{\ell}_2=1$, $\hat{\ell}_3=1$, and $\hat{\gamma}_1=100$.

Notice that we work with the ℓ_i's and σ^2 in doing the calculations and have the same overparametrization here as in the general algorithm. Nevertheless, we converged to the same values for $\hat{s}_1^2$, $\hat{s}_2^2$, $\hat{s}_3^2$, and $\hat{\sigma}_1$ for a variety of different starting values. We also used the general computer algorithm discussed earlier to confirm the final values given in Table 2 to 12 significant digits of agreement in the log-likelihood. To improve the precision of our computations we subtracted the mean from each of the y_{ij} values prior to processing the data. The computations were then performed using double precision arithmetic (as were all of the computations for this paper).

For this example, the balanced ℓ_i algorithm when carefully implemented requires less than 3% of the amount of array storage used by the general algorithm. Furthermore, it requires less than 2% of the amount of computer time used by the general algorithm to achieve the same degree of precision in the results.

We have also had success in modifying the balanced ℓ_i algorithm to efficiently compute REML estimators. The REML estimates for the ℓ_i's are calculated by using equations analogous to (6.14) to (6.16). With λ_1 given by (3.14) these equations are

$$(6.17)\quad \frac{\partial\lambda_1}{\partial\ell_i} = -\frac{(N-1)}{2}a_{ii}+\frac{1}{2\sigma^2}\left\{\sum_{j=1}^{N}\left(\sum_{k=1}^{M}d_{kj}a_{ki}\right)^2\right\},$$

$$(6.18)\quad \frac{\partial^2\lambda_1}{\partial\ell_i\partial\ell_j} = \frac{(N-1)}{2}a_{ij}^2$$

$$-\frac{a_{ij}}{\sigma^2}\left\{\sum_{k=1}^{N}\sum_{\ell=1}^{M}\left(d_{\ell k}a_{\ell i}-\frac{1}{N}\sum_{r=1}^{N}d_{\ell r}a_{\ell i}\right)\sum_{m=1}^{M}d_{mk}a_{mj}\right\},$$

$$(6.19)\quad E\left(\frac{\partial^2\lambda_1}{\partial\ell_i\partial\ell_j}\right) = -\frac{(N-1)}{2}a_{ij}^2,$$

where d is again given by (6.13) with α replaced by (6.11).

Also, the determinant $|THT'|$ which is found in λ_1 can be evaluated via

$$(6.20)\quad |THT'| = \left\{\gamma_1^N\left(\prod_{i=1}^{M}\ell_i\right)^{N-1}/N^M\right\}\cdot|D^{-1}+U_1'T'(TLT')^{-1}TU_1|$$

where the determinant on the right-hand side of (6.20) is a by-product of the W transformation.

REML estimators were also computed for the same set of data used for the ML example. The values .0144, .0438, .2171, and 1.9347 were obtained as estimates of s_1^2, s_2^2, s_3^2, and σ_1^2 respectively.

7. SPECIAL MODELS: UNBALANCED DATA

Although the examples of the model (6.1) appearing in the literature normally deal with balanced data, some provision should realistically be made for missing data values. Suppose that, for notational simplicity, we partition the y vector for the model (6.1) such that

$$(7.1)\quad y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

where y_1 is a q vector corresponding to q missing observations and y_2 is an (n-q) vector of observed data. In a similar manner we partition

$$(7.2)\quad X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix},\quad E_i = \begin{pmatrix} E_{i1} \\ E_{i2} \end{pmatrix},\quad U_1 = \begin{pmatrix} U_{11} \\ U_{12} \end{pmatrix}.$$

$$(7.3)\quad H = \left(\begin{array}{c|c} H_{11} & H_{12} \\ \hline H_{12}' & H_{22} \end{array}\right),\quad H^{-1} = \left(\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{12}' & B_{22} \end{array}\right).$$

The maximum likelihood ℓ_i partials, for the observed data are then given by

$$(7.4)\quad \frac{\partial\lambda}{\partial\lambda_i} = -\frac{1}{2}\,\mathrm{Tr}\{E_{i2}H_{22}^{-1}E_{i2}\}$$

$$+\frac{1}{2\sigma^2}(y_2-X_2\alpha)'H_{22}^{-1}E_{i2}E_{i2}'H_{22}^{-1}(y_2-X_2\alpha),$$

and it is well known that we may express H_{22}^{-1} in (7.4) in terms of the elements of H^{-1} as

$$(7.5)\quad H_{22}^{-1} = B_{22}-B_{12}'B_{11}^{-1}B_{12}.$$

For ML estimation we essentially use the relationship (7.5) to obtain H_{22}^{-1} from the balanced H^{-1} which we already know; but, we do this in a way that ties in nicely with the balanced ℓ_i algorithm and, for a small amount of missing data, requires approximately (1/N)th of the storage required by the general algorithm in containing H_{22}^{-1}. Furthermore, the amount of computation is reduced substantially.

Initially, we store the N diagonal elements of the ijth diagonal blocks of H^{-1} in row major ordering for $i \le j$, in a linear array as

$$(7.6)\quad \overbrace{a_{11},\cdots,a_{11}}^{N},\ \overbrace{a_{12},\cdots,a_{12}}^{N},\cdots,\overbrace{a_{MM},\cdots,a_{MM}}^{N}$$

11 block 12 block MM block

TABLE 1

Summary of Maximization Execution Times[1] and Data Set Characteristics.

Data Set	p	m	c	Execution Time (secs.)		No. of Steps
				BMDQ3V	Cholesky W Trans.	
1	3	8	2	1.63	.77	9
2	3	44	4	59.57	19.19	9
3	1	61	8	173.94	33.69	10

Data 1: Oven Data, Bowker and Lieberman [1], p. 362.

Data 2: Turnip Green Data, Snedecor [13], p. 365.

Data 3: Swine Data, Snedecor [13], p. 360.

[1]The times for data set 1 are from an IBM 370/155. The times for data sets 2 and 3 are from an ITEL Advanced System/5.

TABLE 2

Summary of Calculations for the Balanced Velocity Chronograph Data

Fixed Effects					
	$\hat{\alpha}_1$		$\hat{\alpha}_2$		$\hat{\alpha}_3$
	792.4583333		793.0666666		792.3416666
Step	$\hat{\sigma}^2$	$\hat{\gamma}_1$	$\hat{\ell}_1$	$\hat{\ell}_2$	$\hat{\ell}_3$
0	.0628182294	100.00000000	1.0000000000	1.0000000000	1.000000000
** 1	.0465849312	42.02936857	.2895803462	.6992443331	4.621304986
* 2	.0466757995	37.30136948	.3228890941	.8360312287	4.279427970
3	.0467126746	37.88541045	.2879708705	.8592276851	4.262282065
6	.0467163814	37.96310735	.2843958388	.8612274862	4.260314670
8	.0467163814	37.96311035	.2843958036	.8612274782	4.260314499
10	.0467163814	37.96311035	.2843958036	.8612274782	4.260314499
Step	$\hat{\sigma}_1^2$	$\hat{s}_1^2$	$\hat{s}_2^2$	$\hat{s}_3^2$	λ
0	6.281822936	.0628182294	.0628182294	.0628182294	-35.509269302009
** 1	1.957133243	.0134845548	.0325609062	.2151949916	-27.712724236744
* 2	1.741071242	.0150711066	.0390224258	.1997457213	-27.595715027062
3	1.769728850	.0134518896	.0401368232	.1991025951	-27.593245080587
6	1.773499000	.0132859445	.0402334317	.1990264848	-27.593217095291
8	1.773499142	.0132859428	.0402334313	.1990264770	-27.593217095291
10	1.773499142	.0132859428	.0402334313	.1990264770	-27.593217095291

(**), (*) Fisher scoring: (**) for γ_1 and the ℓ_1's; (*) for the ℓ_1's only

where the a_{ij}'s are given by (6.6). We then modify these values in accordance with the missing data in the following manner.

Consider first one missing value, say y_{ij}. In order to obtain H_{22}^{-1} we need only perform an in place Gauss-Jordan reduction or sweep of H^{-1} given by (6.4), pivoting on the jth diagonal element in the ith diagonal block of H^{-1} (iith block of (7.6)). Only those elements in the jth position in each diagonal block are affected (changed) by this reduction. If we then zero out the jth diagonal element in the ikth and kith blocks of (7.6) for $k=1,\cdots,i$ and $k=i+1,\cdots,M$, respectively, what remains is H_{22}^{-1} arrayed in storage corresponding to the observed data; that is, the row and column corresponding to the missing observation have been replaced by zeroes. This process may be repeated for each additional missing value, which results in the formation of H_{22}^{-1} computed in the positions corresponding to the observed data and zeroes elsewhere.

We also store zeroes in the y vector corresponding to the missing data and modify the appropriate elements of W_o^* to reflect the different frequencies. That is, if y_{ij} is the only missing value, W_o^* would be modified so that the value of the jth diagonal element of $U_i'L^{-1}U_i$ is $\sum_{k\neq i} 1/\ell_k$, the jith element of $U_i'L^{-1}X$ is 0, and the ith diagonal element of $X'L^{-1}X$ is $(N-1)/\ell_i$. Using the above configurations of H_{22}^{-1} and the y vector, the ℓ_i partials or their expected values can then be easily computed in an orderly manner. For example, the quantity $Tr\{E_{i2}'H_{22}^{-1}E_{i2}\}$ in (7.4) is obtained by summing the N elements which have overwritten the iith block of (7.6); the quantity $Tr\{E_{i2}'H_{22}^{-1}E_{j2}E_{j2}'H_{22}^{-1}E_{i2}\}$, applicable to 2nd order ℓ_i partials or their expected values, is obtained by summing the squares of the N elements which have overwritten the ijth block of (7.6). An example of this ML missing data algorithm is given in [9].

For unbalanced data, the variance-covariance matrix of the REML estimators no longer has the sparse structure of H_{22} that allowed for its compacted storage and logical manipulation. We have been unsuccessful in designing an efficient missing data algorithm for these estimators.

REFERENCES

[1] Bowker, A. H. and Lieberman, G. J. (1963). Engineering Statistics, Englewood Cliffs, N.J., Prentice-Hall.

[2] Corbeil, R. R. and Searle, S. R. (1976). Restricted Maximum Likelihood (REML) Estimation of Variance Components in the Mixed Model. Technometrics 18, 31-38.

[3] Downs, B. W. (1977). Optimization Algorithms for the Mixed Analysis of Variance Model. Unpublished M.S. Thesis, University of Rhode Island, Kingston, R.I.

[4] Grubbs, F. E. (1948). On Estimating Precision of Measuring Instruments and Product Variability. J. Amer. Statist. Assoc. 43, 243-264.

[5] Grubbs, F. E. (1973). Errors of Measurement, Precision, Accuracy and the Statistical Comparison of Measuring Instruments. Technometrics 15, 53-66.

[6] Hartley, H. O. and Rao, J. N. K. (1967). Maximum Likelihood Estimation for the Mixed Analysis of Variance Model. Biometrica, 54, 93-108.

[7] Hemmerle, W. J. and Hartley, H. O. (1973). Computing Maximum Likelihood Estimates for the Mixed A.O.V. Model Using the W Transformation. Technometrics 15, 819-831.

[8] Hemmerle, W. J. and Lorens, J. A. (1976). Improved Algorithm for the W-Transform in Variance Component Estimation, Technometrics 18, 207-212.

[9] Hemmerle, W. J., and Downs, B. W. (1978). Non-Homogeneous Variances in the Mixed A.O.V. Model: Maximum Likelihood Estimation, in Contributions to Survey Sampling and Applied Statistics--Papers in Honor of H. O. Hartley, N.Y., Academic Press.

[10] Jennrich, R. I. and Sampson, P. F. (1976). Newton-Raphson and Related Algorithms for Maximum Likelihood Variance Component Estimation. Technometrics 18, 11-18.

[11] Jennrich, R. I. and Sampson, P. F. (1977). General Mixed Model Analysis of Variance. BMDP-77, Biomedical Computer Programs P-Series, University of California Press.

[12] Miller, J. J. (1973). Asymtotic Properties and Computation of Maximum Likelihood Estimates in the Mixed Model of the Analysis of Variance. Technical Report No. 12, Department of Statistics, Stanford University.

[13] Snedecor, G. W. (1956). Statistical Methods, 5th ed., Ames, Iowa State University Press.

*Research supported in part by NSF Grants DCR74-15331 and MCS74-15331 A01.

Some Problems Faced in Making a Variance Component Algorithm into a General Mixed Model Program*

Robert I. Jennrich and Paul F. Sampson

University of California at Los Angeles

ABSTRACT

Three algorithmic problems encountered in producing a general mixed model analysis of variance and covariance program BMDP3V from a general Newton-Raphson algorithm for estimating mean and variance components are discussed. The first concerns the implementation of a simple switch to allow the program to perform both maximum likelihood and restricted maximum likelihood analyses. The second involves a resolution of the sum to zero problem for interaction terms in the mixed model. The third involves a method of dealing explicitly with boundary constraints for the variance components and more generally with constrained optimization.

*This research was supported in part by National Institutes of Health Grant RR-3.

1. INTRODUCTION

Several years ago the authors proposed a Newton-Raphson algorithm [6] for estimating mean and variance components for a general mixed analysis of variance model. Since then the algorithm has been developed into a statistical program BMDP3V which is part of the BMDP series [7]. We will discuss some of the algorithmic problems encountered in making a basic algorithm into a program.

The first of these involves the ML (maximum likelihood) versus REML (restricted maximum likelihood [1]) debate. On the one hand asymptotic considerations and some simulation studies [2] and [10] seem to suggest that ML estimates, particularly of variance components, may be superior to REML estimates. There are arguments on the other side, however, perhaps the most persuasive being Searle's conjecture [1] that in the balanced case REML and ANOVA (analysis of variance) estimates are identical. This represents a distinct practical advantage to program developers who are well aware of the problems that arise when results vary from program to program. Rather than attempting to resolve the ML-REML issue, we did what software developers usually do when they don't know what to do. We made the choice a user option. Algorithmically this is quite easy to do by means of an ML-REML switch which converts a Newton-Raphson ML algorithm into a Newton-Raphson REML algorithm. To our knowledge a Newton-Raphson algorithm for REML estimation has not been discussed elsewhere.

The second problem we will discuss is the "mixed model sum to zero" problem. Many authors [3], [4], [6], e.g., have been willing to model the interaction components in a two-way mixed model as a set of uncorrelated variables with constant variance while others, particularly Kempthorne [8], argue rather convincingly that from sampling considerations these components should sum to zero over the fixed index and hence be correlated. While the debate here may be of greater

interest than the issue, it nevertheless poses a problem to the program developer. The "make it a user option" solution works again as soon as one finds a convenient parameterization for the "sum to zero" model in terms of the fixed and random coefficients model. We propose such a parameterization.

The third and final problem addressed concerns the handling of boundary constraints. In the present context, this is the constraint that proper variance components are non-negative. In developing algorithms we and others have been a little cavalier about this problem, more or less setting it aside while more basic issues were addressed. In a production program we need to be more careful. Some (e.g., Hemmerle and Hartley, [4]) have dealt with boundary constraints by choosing a parameterization which eliminates them. Others, like ourselves [6], have dealt with them explicitly. The latter approach which we will discuss here is fairly simple and seems to be preferable. Moreover, the techniques generalize easily to other applications of constrained optimization in statistical computing.

2. THE ML-REML SWITCH

Consider the standard fixed and random coefficients model

$$y = X\alpha + U_1 b_1 + \ldots + U_c b_c + e \qquad (1)$$

where

- y is a vector of n observations
- X is a known n by p matrix
- α is a p-vector of unknown parameters
- U_i is a known n by q_i matrix
- b_i is a q_i-vector of random values from $n(0,\sigma_i^2)$
- e is an n-vector of random values from $n(0,\sigma^2)$.

The random vectors $b_1,\ldots,b_c$, and e are assumed independent. It follows at once that y has a multivariate normal density

$$f(y) = (2\pi)^{-n/2}|\Sigma|^{-1/2}\exp(-\tfrac{1}{2}(y - X\alpha)'\Sigma^{-1}(y - X\alpha)) \qquad (2)$$

where

$$\Sigma = \sigma_1^2 U_1 U_1' + \ldots + \sigma_c^2 U_c U_c' + \sigma^2 I. \qquad (3)$$

Let

$$\lambda = \log f(y) \qquad (4)$$

denote the corresponding log-likelihood, let

$$\theta = (\sigma_1^2,\ldots,\sigma_c^2,\sigma^2) \qquad (5)$$

denote the entire set of variance components, let

$$U = [U_1,\ldots,U_c] \qquad (6)$$

be the design matrices for all of the random components except e in the basic model (1), and finally let

$$C = \begin{pmatrix} X'X & X'U & X'y \\ U'X & U'U & U'y \\ y'X & y'U & y'y \end{pmatrix} \qquad (7)$$

denote a partitioned matrix containing the indicated sums of cross products. The log-likelihood λ is clearly a function of y, X, U, α, and θ. Actually, however, its dependence on the "data" U,X, and y is only through C. More precisely we have the

ML Sufficiency Theorem: The log-likelihood λ is a function of α, θ, C, and n.

Proof: From (2) and (4)

$$\lambda = -\frac{n}{2}\log 2\pi - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(y - X\alpha)'\Sigma^{-1}(y - X\alpha). \qquad (8)$$

From (3) and (6)

$$\Sigma = U\Delta^2 U' + \sigma^2 I \qquad (9)$$

where Δ^2 is a diagonal matrix with diagonal elements $\sigma_1^2,\ldots,\sigma_c^2$. From (9) and the Bartlett identity,

$$\Sigma^{-1} = \sigma^{-2}I - \sigma^{-4}U\Delta(I + \Delta U'U\Delta)^{-1}\Delta U'. \qquad (10)$$

From (5), (7), and (10) it is clear that

$$(y - X\alpha)'\Sigma^{-1}(y - X\alpha) \qquad (11)$$

is a function of α, θ, and C. From (9)

$$|\Sigma| = \sigma^{2n}|I + \sigma^{-2}\Delta U'U\Delta| \qquad (12)$$

which is clearly a function of θ, C, and n. From (8), (11), and (12), λ is a function of α, θ, C, and n.

We may summarize the ML Sufficiency Theorem by writing

$$\lambda = L(\alpha,\theta,C,n). \qquad (13)$$

One consequence of the theorem in the present context is that algorithms, like NR (Newton-Raphson), which maximize λ with respect to α and θ must have an implementation which computes the length n inner products found in C once and for all; not requiring recomputation from iteration to iteration. Our NR algorithm uses such an implementation, which is better known as the W-transform technology of Hemmerle and Hartley [4]. A similar theorem applies to REML estimation to which we now turn.

Let $y.X$ denote the vector of residuals resulting from least squares regression of y on X. A simple definition of REML estimation is that it is maximum likelihood estimation based on

$$\tilde{y} = y.X \qquad (14)$$

rather than on y itself. The REML model is

$$\tilde{y} = \tilde{U}_1 b_1 + \ldots + \tilde{U}_c b_c + \tilde{e} \qquad (15)$$

where $\tilde{U}_r = U_r.X$ for $r = 1,\ldots,c$ and $\tilde{e} = e.X$. The covariance matrix for $\tilde{y}$ is in general singular and has the form

$$\tilde{\Sigma} = \sigma_1^2\tilde{U}_1\tilde{U}_1' + \ldots + \sigma_c^2\tilde{U}_c\tilde{U}_c' + \sigma^2\tilde{I} \qquad (16)$$

where $\tilde{I}$ is the perpendicular projection onto the orthogonal complement, $R(X)^\perp$, of the column space of X. While $\tilde{y}$ does not in general have a density on Euclidean n-space, it does have a well defined density with respect to Lebesgue measure on $R(X)^\perp$ given by

$$\tilde{f}(\tilde{y}) = (2\pi)^{-\tilde{n}/2}|\tilde{\Sigma}|^{-1/2}\exp(-\tfrac{1}{2}\tilde{y}'\tilde{\Sigma}^{-1}\tilde{y}) \qquad (17)$$

where $\tilde{n} = n - p$ and $\tilde{\Sigma}$ is to be viewed as a non-singular linear transformation from $R(X)^\perp$ onto $R(X)^\perp$. This guarantees that its inverse is defined and its determinant is different from zero.

Letting

$$\tilde{\lambda} = \log \tilde{f}(\tilde{y}), \qquad (18)$$

$$\tilde{U} = [\tilde{U}_1,\ldots,\tilde{U}_c] \qquad (19)$$

and

$$\tilde{C} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \tilde{U}'\tilde{U} & \tilde{U}'\tilde{y} \\ 0 & \tilde{y}'\tilde{U} & \tilde{y}'\tilde{y} \end{pmatrix} \qquad (20)$$

we have the

REML Sufficiency Theorem: The REML log-likelihood $\tilde{\lambda}$ is a function of θ, $\tilde{C}$, and $\tilde{n}$ given by

$$\tilde{\lambda} = L(0,\theta,\tilde{C},\tilde{n}). \qquad (21)$$

The function L does not accidently have a tilda missing. It is precisely the same function whose existence is asserted in the ML Sufficiency Theorem. Since a proof of the REML theorem is almost line for line the same as that for the ML theorem, it will not be given here. The importance of the REML theorem in the present context is that algorithms which one has produced to compute λ and its derivatives may also be used to compute $\tilde{\lambda}$ and its derivatives by simply replacing α, C, and n by 0, $\tilde{C}$, and $\tilde{n}$. Since $\tilde{C}$ as given by (20) can be obtained from C as given by (7) by Gauss-Jordan pivoting [5] on the diagonal elements of $X'X$ and then setting the corresponding rows and columns to zero, it is a very simple matter to implement the NR-REML algorithm:

$$\Delta\theta = -\left(\frac{d^2\tilde{\lambda}}{d\theta\, d\theta}\right)^{-1}\frac{d\tilde{\lambda}}{d\theta} \qquad (22)$$

if one has already implemented the corresponding NR-ML algorithm.

This completes the story with regard to REML variance component estimation. A program developer, of course, also needs to consider REML estimation of α and the estimation of standard errors for all estimates. These considerations require further theorems, but only

minor program modifications given that one is beginning with a complete NR-ML program. Without going into details we will simply summarize the modifications required for a complete ML-REML switch:

(i) Replace (α,C,n) by $(0,\tilde{C},\tilde{n})$ and NR iterate to convergence. This gives $\tilde{\theta}$.

(ii) Make one Fisher scoring step starting at $(\alpha,\theta) = (0,\tilde{\theta})$. This gives

$$\tilde{\alpha},\ \widehat{\operatorname{cov}}\,\tilde{\alpha},\ \widehat{\operatorname{cov}}\,\tilde{\theta}.$$

Ignore the scoring update for θ.

Here $\tilde{\alpha}$ and $\tilde{\theta}$ denote REML estimates and $\widehat{\operatorname{cov}}\,\tilde{\alpha}$ and $\widehat{\operatorname{cov}}\,\tilde{\theta}$ denote information theory estimates of their sampling covariance matrices.

Since there is to our knowledge only one published algorithm for REML estimation in the context of the general model (1), that of Corbeil and Searle [1], we should point out how the algorithm just discussed is related to theirs. The Corbeil and Searle algorithm is based on a slightly different parameterization. They use α and σ^2 as we have, but in place of the variance components σ_i^2 they use relative variance components

$$\gamma_i^2 = \sigma_i^2/\sigma^2; \qquad i = 1,\ldots,c. \tag{23}$$

Their algorithm has two phases, a $\gamma^2 = (\gamma_1^2,\ldots,\gamma_c^2)$ phase and a σ^2 phase. During the γ^2 phase σ^2 is held fixed and a NR step is applied to $\tilde{\lambda}$ viewed as a function of γ^2. During the σ^2 phase, γ^2 is held fixed and $\tilde{\lambda}$ is maximized with respect to σ^2. This produces a see-saw algorithm similar to that used by Hemmerle and Hartley [4] for ML estimation. Unfortunately, the see-saw destroys the quadratic convergence rate one would have obtained if NR steps were applied to all of the parameters in $\tilde{\lambda}$ as we have done. The choice of parameterizing in terms of variance components σ_i^2 or relative variance components γ_i^2 seems to be of secondary importance. We have tried both and, primarily for conceptual reasons and partly for performance reasons, prefer to work directly with the variance components σ_i^2 rather than the relative variances γ_i^2. When the REML or ML estimate of σ^2 is small, the relative variance formulation seems to work poorly.

3. THE MIXED MODEL SUM TO ZERO OPTION

An example should be sufficient here. Consider a classical mixed model of the form

$$y_{ijk} = \nu + \alpha_i + b_j + c_{ij} + e_{ijk} \tag{24}$$

where $i = 1,\ldots,I$; $j = 1,\ldots,J$; and $k = 1,\ldots,K_{ij}$. Here the y_{ijk} denote observations, ν and the α_i denote fixed components, and the b_j, c_{ij}, and e_{ijk} denote random components. It is customary to assume that

$$\Sigma_i \alpha_i = E\, b_j = E\, c_{ij} = E\, e_{ijk} = 0 \tag{25}$$

for all i, j, and k. As mentioned earlier it is also frequently assumed that the

$$b_j,\ c_{ij},\ e_{ijk} \text{ are uncorrelated} \tag{26}$$

for all i, j, and k, and that these quantities have homogeneous variances, i.e.,

$$\operatorname{var} b_j = \sigma_b^2, \quad \operatorname{var} c_{ij} = \sigma_c^2, \quad \operatorname{var} e_{ijk} = \sigma^2 \tag{27}$$

for all i, j, and k. The "sum to zero" debate is focused on the interaction term c_{ij}. Proponents of the sum to zero position, including the authors, argue that from sampling considerations it is natural to assume that

$$\Sigma_i\, c_{ij} = 0, \tag{28}$$

i.e., that the c_{ij} sum to zero over the "fixed" index i. The homogeneity of variance assumption for the c_{ij} then takes the form

$$\operatorname{var} c_{ij} = (1 - 1/I)\sigma_c^2 \tag{29}$$

and

$$\operatorname{cov}(c_{ij},c_{i'j}) = -\sigma_c^2/I \tag{30}$$

for all i, i', and j. All other model assumptions are as before. The only algorithmic problem with the

sum to zero assumption is that under it (24) cannot be as simply formulated as a fixed and random coefficients model of the form (1) as it can when the c_{ij} are assumed uncorrelated. This is because correlated c_{ij} cannot play the role of random coefficients in (1). A simple change of variable, however, will do the trick. Let

$$c_{ij} = \sum_{r=1}^{I-1} \sum_{s=1}^{J} p_{rs}^{(I)} \delta_{js} \tilde{c}_{rs} \qquad (31)$$

where $p_{rs}^{(I)}$ is the value at s of the orthogonal polynomial of degree r on $\{1,2,\ldots,I\}$, δ_{js} is the Kronecker delta, and the $\tilde{c}_{rs}$ are uncorrelated random variables satisfying

$$E\,\tilde{c}_{rs} = 0, \qquad \text{var}\,\tilde{c}_{rs} = \sigma_c^2 \qquad (32)$$

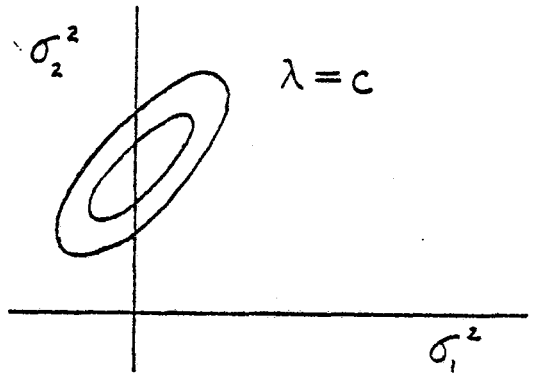
for all r and s. Under these conditions it is easy to show that the c_{ij} defined by (31) satisfy the sum to zero conditions (28), (29), and (30). Moreover, the $\tilde{c}_{rs}$ may be used as random coefficients when formulating (24) in terms of the general fixed and random coefficients model (1). The orthogonal polynomials $p_{rs}^{(I)}$ may be replaced by any orthonormal set of functions on the index values $\{1,\ldots,I\}$ which sum to zero over these values. Clearly, straightforward generalizations of (31) may be used to deal with more complex manifestations of the sum to zero problem. In a general setting this involves analysis of variance terms with crossed and nested indices which may be fixed or random. The terms should sum to zero over all indices which are fixed and crossed and should satisfy homogeneity of variance assumptions analogous to (29) and (30). All of this may be accomplished through parameterizations similar to (31).

4. BOUNDARY CONSTRAINTS

As observed in the introduction, the boundary constraint problem in the present context concerns the requirement that

$$\sigma_i^2 \geq 0 \quad \text{and} \quad \sigma^2 \geq 0 \qquad (33)$$

for $i = 1,\ldots,c$. One resolution of this problem is to choose σ and the σ_i as the basic parameters rather than σ^2 and the σ_i^2. When this is done the constraints (33) are automatically satisfied. Since the log-likelihoods λ and $\tilde{\lambda}$ are functions of σ^2 and the σ_i^2, however, it seems natural to write algorithms in terms of these reduced parameters, i.e., the squares, and deal with the constraints (33) directly. In our experience both approaches work, but the latter approach works better. One possible theoretical reason is depicted in the following figure:



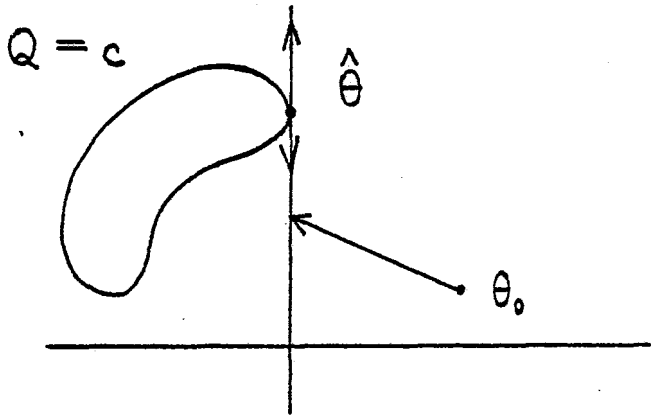
When the unconstrained maxima of λ or $\tilde{\lambda}$ lie exactly on a boundary and these log-likelihoods are viewed as functions of the σ_i, their Hessians and information matrices must be singular. As the figure suggests, however, these matrices may be well behaved when λ and $\tilde{\lambda}$ are viewed as functions of the σ_i^2. Clearly singular Hessians and information matrices can cause algorithmic problems as can nearly singular Hessians and information matrices arising from unconstrained maxima which lie near boundaries.

Independent of these issues, the ability to deal explicitly with constraints, particularly coordinate inequality constraints, clearly represents an important and fundamental technology in statistical computing which probably has received less attention than it deserves from statistical programmers.

The basic problem here is

$$\begin{aligned} &\text{min:} \quad Q(\theta) \\ &\text{given:} \quad \theta \geq 0. \end{aligned} \tag{34}$$

We have stated it as a minimization rather than a maximization problem because this is customary in the programming literature as well as most of the statistical literature outside of maximum likelihood estimation. The basic idea of the algorithm we are about to discuss is depicted in the figure below:



The algorithm will begin with a feasible value θ_0. When active constraints are involved it will typically move to a boundary and then search along the boundary until a constrained minimum $\hat{\theta}$ is found. Simple boundary projection (replacing negative components of θ by zero whenever they occur) will not always work here. We mention this only because there exist examples in the variance components literature where it has been used and has led to the publication of results which do not represent constrained maxima. While with more careful formulation boundary projection can be made to work, it does not seem to lead to algorithms as simple as those of the form we will discuss.

We begin with the following

Basic Algorithm:

(i) Given a feasible value θ use the Quadratic Programming Algorithm below to find a feasible direction $\Delta\theta$. If $\Delta\theta = 0$, stop.

(ii) Use the Partial Step Algorithm below to find an $\alpha \in [0,1]$ such that the step $\alpha\Delta\theta$ leads to a reduction in $Q(\theta)$.

(iii) Replace θ by $\theta+\alpha\Delta\theta$ and go to (i).

One may prove that such an algorithm will converge to a local constrained minimum $\hat{\theta}$ if the initial value is close enough to $\hat{\theta}$, if the Hessian of $Q(\theta)$ exists and is positive definite at $\hat{\theta}$ and if the quadratic programming and partial step sub-algorithms have appropriate properties (see, e.g., [9]). Of the quadratic programming algorithm we will demand that $\theta+\Delta\theta$ be feasible, that $(dQ/d\theta)\Delta\theta < 0$ when θ is not a local constrained minimum, and that $\Delta\theta$ be a continuous function of θ. Of the partial step algorithm we demand that it reduce $Q(\theta)$ by at least $\delta \cdot (dQ/d\theta)\Delta\theta$ for some fixed $\delta > 0$ whenever $(dQ/d\theta)\Delta\theta \neq 0$. Here δ may depend on the function $Q(\theta)$ but not on θ itself.

The difficult part is producing continuous feasible directions $\Delta\theta$. For this we introduce an asscciated quadratic function

$$Q^*(\Delta\theta) = b\Delta\theta + \tfrac{1}{2}\Delta\theta' A\Delta\theta \tag{35}$$

where $b = dQ/d\theta$ and A is positive definite. The choice of A depends on the basic algorithm employed. In our previous discussion, e.g., A might be minus the Hessian of λ when working with the NR algorithm, or the information matrix $\mathcal{J}(\theta)$ when implementing the Fisher scoring algorithm. Given the associated quadratic function (35), one may produce a feasible $\Delta\theta$ by solving the associated quadratic programming problem

$$\begin{aligned} &\text{min:} \quad Q^*(\Delta\theta) \\ &\text{given:} \quad \Delta\theta \geq -\theta. \end{aligned} \tag{36}$$

This will give a $\Delta\theta$ with all of the properties demanded above provided $dQ/d\theta$ and A are continuous functions of θ.

The natural tabloid to consider in conjunction with the associated quadratic programming problem (36) is displayed on the left below:

$$\begin{pmatrix} A & -b' \\ -b & 0 \end{pmatrix} \rightarrow \begin{pmatrix} * & u \\ u' & \Delta Q^* \end{pmatrix}. \qquad (37)$$

Let the tabloid on the right be the result of applying Gauss-Jordan pivots [5] to the diagonal elements of A corresponding to those components θ_i of θ which are not critical, i.e., not equal to zero. Let the vector u be divided into two parts u_1 and u_2 corresponding to the pivoted and unpivoted diagonals of A respectively.

It follows from the Kuhn-Tucker theory [9] that if $u_1 = 0$ and $u_2 \leq 0$, then $\Delta\theta = 0^*$ is a solution to the associated quadratic problem. The following algorithm finds a $\Delta\theta$ such that $\widetilde{\Delta\theta} = 0$ is such a solution to the problem

$$\begin{aligned} \text{min:} \quad & \widetilde{Q}(\widetilde{\Delta\theta}) = Q^*(\Delta\theta + \widetilde{\Delta\theta}) \\ \text{given:} \quad & \widetilde{\Delta\theta} \geq -(\theta + \Delta\theta). \end{aligned} \qquad (38)$$

This implies that $\Delta\theta$ itself is a solution to the original associated quadratic programming problem (36).

The Quadratic Programming Algorithm:

(i) Form the tabloid on the left in (37). Let C be the set of subscripts i for which $\theta_i = 0$. For all $i \notin C$ perform a Gauss-Jordan pivot on the corresponding diagonal element of the tabloid. Set $\Delta\theta = 0$.

(ii) Find the largest $\alpha \in [0,1]$ such that

$$\theta_i + \Delta\theta_i + \alpha u_i \geq 0, \qquad i \notin C$$

where u_i denotes the i-th element in the last column of the tabloid.

(iii) For each $i \notin C$, replace $\Delta\theta_i$ by $\Delta\theta_i + \alpha u_i$ and u_i by $u_i - \alpha u_i$ and pivot all newly critical diagonals of the tabloid, i.e., all for which $\theta_i + \Delta\theta_i = 0$, $i \notin C$. add the corresponding indices to C.

(iv) If $u_i \neq 0$ for some $i \notin C$, go to (ii).

(v) If $u_i > 0$ for some $i \in C$ pivot one corresponding diagonal element of the tabloid and go to (ii). Otherwise stop.

One can show that if θ is feasible and A is positive definite, the Quadratic Programming Algorithm will converge to a solution $\Delta\theta$ of the associated quadratic programming problem (36) in a finite number of steps. While the algorithm may at first look a little complicated it really represents a fairly minor modification to the pivoting operations one would perform in the unconstrained case. Under normal operation, after the first few steps of the Basic Algorithm θ will generally lie on the appropriate boundary and the steps of the Quadratic Programming Algorithm will be executed only once. Returns to step (ii) are made only when a boundary change is required.

The partial stepping algorithm will require the quantity $\Delta Q = (dQ/d\theta)\Delta\theta$. This may be obtained from the lower right hand corner of the tabloid used in the Quadratic Programming Algorithm at the time of exit from the algorithm.

The Partial Step Algorithm:

(i) Find the largest α in the sequence $\alpha = 1, 1/2, 1/4, \ldots$ such that

$$Q(\theta + \alpha\Delta\theta) < Q(\theta) + \alpha\Delta Q/10 \qquad (39)$$

(ii) Stop.

That such an α exists follows from the fact that the $\Delta\theta$ in the call from the Basic Algorithm is nonzero and minimizes $Q^*(\Delta\theta)$ and from the fact that $Q^*(\Delta\theta)$ and $Q(\theta+\Delta\theta)$ have the same gradient at $\Delta\theta = 0$. In order to guarantee that the α produced by the Partial Step Algorithm are bounded below by some $\delta > 0$, it is sufficient to assume that the A used in the Quadratic Programming Algorithm are bounded away from singularity; to assume, e.g., that the smallest eigen-value of A is greater than some fixed positive value which may depend on the function $Q(\theta)$ but not on θ itself.

Clearly the number 10 in the Partial Step Algorithm is arbitrary. Any value greater than or equal

to 1 will do. A large value like 10 increases the chances that one will use the complete step $\Delta\theta$ in the Basic Algorithm and thus eliminate the expense of computing $Q(\theta + \alpha\Delta\theta)$ for more than one value of α. Clearly the step-halving sequence of values $\alpha = 1, 1/2, 1/4, \ldots$, is also arbitrary. It may be replaced by some other decreasing sequence, or a sequence based on an analytic, e.g., quadratic, search.

This completes our discussion of handling boundary constraints of the form $\theta \geq 0$. The techniques may be easily generalized to handle coordinate inequality constraints of the form

$$a \leq \theta \leq b \qquad (40)$$

and with some modification of the tabloid (37), to handle equality and inequality constraints of the form

$$H\theta = c, \qquad a \leq \theta \leq b. \qquad (41)$$

5. COMMENT

A number of important and interesting problems have not been addressed. For example, how should the general mixed model be formulated in the first place? We have used the standard fixed and random coefficients model, but possibly a more general multivariate model is needed. We have employed normal theory ML and REML estimates, but perhaps more robust estimates are needed. In spite of the W-transform methods of Hemmerle and Hartley [4], general purpose ML and REML programs for the unbalanced mixed model use a lot of computer storage placing rather severe size limitations on the problems that can be handled especially if primary interest is in variance component estimation. What, if anything, can be done about this? Even if we admit that for now we need to use asymptotic theory for tests and confidence intervals, which of the many asymptotic approaches do we use? These are some of the things that continue to bother us as program developers.

REFERENCES

[1] Corbeil, R. R. and S. R. Searle (1976). Restricted maximum likelihood (REML) estimation of variance components in the mixed model. Technometrics 18, 31-38.

[2] Corbeil, R. R. and S. R. Searle (1976). A comparison of variance component estimators. Biometrics 32, 779-791.

[3] Hartley, H. O. and J. N. K. Rao (1967). Maximum likelihood estimation for the mixed analysis of variance model. Biometrika 54, 93-108.

[4] Hemmerle, W. J. and H. O. Hartley (1973). Computing maximum likelihood estimates for the mixed A.O.V. model using the W transform. Technometrics 15, 819-831.

[5] Jennrich, R. I. and P. F. Sampson (1968). Application of stepwise regression to nonlinear estimation. Technometrics 10, 63-72.

[6] Jennrich, R. I. and P. F. Sampson (1976). Newton-Raphson and related algorithms for maximum likelihood variance component estimation. Technometrics 18, 11-17.

[7] Jennrich, R. I. and P. F. Sampson (1977). BMDP3V, General mixed model analysis. BMDP Biomedical Computer Programs, Dixon, W. J. and M. B. Brown, editors, University of California Press, Berkeley, 581-598.

[8] Kempthorne, O. (1975). Fixed and mixed models in the analysis of variance. Biometrics 31, 473-486.

[9] Luenberger, D. G. (1973). Introduction to Linear and Nonlinear Programming. Addison-Wesley, Reading, Massachusetts.

[10] Thompson, W. O. and R. L. Anderson (1975). A comparison of designs and estimators for the two-stage nested random model. Technometrics 17, 37-44.

A SUMMARY OF RECENTLY DEVELOPED METHODS OF ESTIMATING VARIANCE COMPONENTS*

S. R. Searle
Biometrics Unit, Cornell University
Ithaca, New York

Abstract

The recently developed methods of estimating variance components, namely ML, REML, MINQUE, I-MINQUE and MIVQUE are presented in summary form in a uniform notation. Relationships between the methods are shown. The "mixed model equations" and the dispersion-mean model are also given.

* Paper No. BU-338 in the Biometrics Unit, Cornell University.

1. INTRODUCTION

Variance components estimation was, for several decades, the poor step-child of analysis of variance, but in recent years the subject has generated quite widespread interest. Until ten years ago, methods of estimation were based on equating sums of squares to their expected values, as first proposed by Daniels [1939] and Winsor and Clarke [1940]. For balanced data (having equal numbers of observations in the subclasses), this method involves sums of squares associated with traditional analysis of variance; its use in a variety of models was given in Anderson and Bancroft [1952], and useful minimum variance properties were derived by Graybill and co-workers in the late 1950's and early 1960's, e.g., Graybill and Wortham [1956] and Graybill and Hultquist [1961]. For unbalanced data (having unequal numbers of observations in the subclasses, possibly with some or many empty subclasses), Henderson [1953] is a landmark paper with its three methods of estimation based on the same principle, equating quadratic forms to their expected values. Succeeding years saw expansion and explanation of these methods together with exploration of their properties, but there were no really new developments until Hartley and Rao [1967] described maximum likelihood procedures - based, as is so often the case, on normality assumptions. Since then there has been a whole host of new methods, not only ML, but REML, MINQUE, I-MINQUE, and MIVQUE - and doubtless some other alphabetic horrors also. In addition there are peripheral topics tangential to computing techniques - such as Henderson's MME's (mixed model equations) and the Dispersion-mean model suggested by Pukelsheim [1976]. As foundation for all this there is a large corpus of matrix algebra, there are numerous notations that look sufficiently alike to add the traditional amount of confusion and, hanging like a thunder cloud over everything, are numerical and computing problems involved with very large data sets, sparse matrices, and the solving of non-linear equations subject to non-linear (non-negativity) constraints.

Obviously, it would take a tome to deal thoroughly with all three aspects of the subject: description of each method, details of the underlying algebra, and the computing algorithms. Attention is confined here to just one thing: description of the methods; and to avoid excessive length, the presentation is given somewhat in note form. The prime purpose is to give, in summary form, the basic rationale and methodology for each of the estimation procedures considered; and to do this with a unifying notation and to show relationships between the methods. There are voluminous details of underlying algebra, and equivalent expressions for each method, which will all be available in Searle and Quaas [1978]; and computing algorithms are left to others.

2. THE MODEL

2.1. Equation of the model

The general linear model is represented as

$$\underset{\sim}{y} = \underset{\sim\sim}{X}\underset{\sim}{\alpha} + \underset{\sim\sim}{Z}\underset{\sim}{b} + \underset{\sim}{e} = \underset{\sim\sim}{X}\underset{\sim}{\alpha} + \sum_{i=1}^{c} \underset{\sim\sim}{Z}_i\underset{\sim}{b}_i + \underset{\sim}{e} \qquad (1)$$

where

$\underset{\sim}{y}_{N\times 1}$ is a vector of N observations,

$\underset{\sim\sim}{X}_{N\times p}$ is a known matrix, of rank $p^* \leq p < N$,

$\underset{\sim}{\alpha}_{p\times 1}$ is a vector of p fixed effects parameters,

$\underset{\sim}{Z}_{N\times q}$ is a known matrix,

$\underset{\sim}{b}_{q\times 1}$ is a vector of random effects, and

$\underset{\sim}{e}_{N\times 1}$ is a vector of residual errors.

The second equality in (1) comes from the partitioning

$$\underset{\sim}{b}' = [\underset{\sim}{b}_1' \; \underset{\sim}{b}_2' \cdots \underset{\sim}{b}_c'] \quad \text{and} \quad \underset{\sim}{Z} = [\underset{\sim}{Z}_1 \; \underset{\sim}{Z}_2 \cdots \underset{\sim}{Z}_c] \tag{2}$$

where $\underset{\sim}{b}_i$ has order $q_i \times 1$ and is the vector of q_i effects corresponding to q_i levels of the i^{th} random factor (main effect or interaction factor) in the model, with $q = \sum_{i=1}^{c} q_i$.

2.2. Distributional properties

$$E(\underset{\sim}{b}_i) = \underset{\sim}{0}, \quad E(\underset{\sim}{e}) = \underset{\sim}{0}, \quad \text{and} \quad E(\underset{\sim}{y}) = \underset{\sim}{X}\underset{\sim}{\alpha} \tag{3}$$

$$\text{var}(\underset{\sim}{b}_i) = \sigma_i^2 \underset{\sim}{I}_{q_i}, \quad \text{cov}(\underset{\sim}{b}_i, \underset{\sim}{b}_j') = \underset{\sim}{0}, \; i \neq j,$$

and (4)

$$\text{var}(\underset{\sim}{b}) = \underset{\sim}{D} = \text{diag}\{\sigma_1^2 \underset{\sim}{I}_{q_1} \cdots \sigma_c^2 \underset{\sim}{I}_{q_c}\}$$

$$\text{var}(\underset{\sim}{e}) = \underset{\sim}{R}, \quad \text{cov}(\underset{\sim}{b}_i, \underset{\sim}{e}') = \underset{\sim}{0},$$

and (5)

$$\text{var}(\underset{\sim}{y}) = \underset{\sim}{V} = \underset{\sim}{Z}\underset{\sim}{D}\underset{\sim}{Z}' + \underset{\sim}{R} .$$

It is customary to have $\underset{\sim}{R} = \sigma_0^2 \underset{\sim}{I}$.

2.3. Ancillary notation

$$\underset{\sim}{H} = (1/\sigma_0^2)\underset{\sim}{V} \quad \text{and} \quad \underset{\sim}{\sigma}^{2\prime} = [\sigma_1^2 \cdots \sigma_c^2] \tag{6}$$

$$\underset{\sim}{b}_0 = \underset{\sim}{e}, \; q_0 = N, \; \underset{\sim}{Z}_0 = \underset{\sim}{I}_N, \text{ and } \dot{\underset{\sim}{\sigma}}^{2\prime} = [\sigma_0^2 \; \sigma_1^2 \cdots \sigma_c^2] \tag{7}$$

$$\dot{\underset{\sim}{D}} = \text{diag}\{\sigma_0^2 \underset{\sim}{I}_{q_0} \; \sigma_1^2 \underset{\sim}{I}_{q_1} \cdots \sigma_c^2 \underset{\sim}{I}_{q_c}\}; \; \dot{\underset{\sim}{Z}} = [\underset{\sim}{Z}_0 \; \underset{\sim}{Z}_1 \cdots \underset{\sim}{Z}_c] \tag{8}$$

$$\underset{\sim}{V} = \dot{\underset{\sim}{Z}}\dot{\underset{\sim}{D}}\dot{\underset{\sim}{Z}}' \tag{9}$$

$$\underset{\sim}{P} = \underset{\sim}{V}^{-1} - \underset{\sim}{V}^{-1}\underset{\sim}{X}(\underset{\sim}{X}'\underset{\sim}{V}^{-1}\underset{\sim}{X})^{-}\underset{\sim}{X}'\underset{\sim}{V}^{-1} \tag{10}$$

3. ML: Maximum Likelihood

3.1. Rationale

Assume normality of the random effects (the b's and e's) with first and second moments as in (3), (4) and (5), and maximize the logarithm of the likelihood of $\underset{\sim}{y}$:

$$L_y = -\tfrac{1}{2}N \log 2\pi - \tfrac{1}{2}\log|\underset{\sim}{V}| - \tfrac{1}{2}(\underset{\sim}{y} - \underset{\sim}{X}\underset{\sim}{\alpha})'\underset{\sim}{V}^{-1}(\underset{\sim}{y} - \underset{\sim}{X}\underset{\sim}{\alpha}) \tag{11}$$

3.2. Method

Maximizing L_y with respect to $\underset{\sim}{\alpha}$ and to the σ^2's implicit in $\underset{\sim}{V}$ leads to equations in $\tilde{\underset{\sim}{\alpha}}$ and $\tilde{\underset{\sim}{V}}$:

$$\underset{\sim}{X}'\tilde{\underset{\sim}{V}}^{-1}\underset{\sim}{X}\tilde{\underset{\sim}{\alpha}} = \underset{\sim}{X}'\tilde{\underset{\sim}{V}}^{-1}\underset{\sim}{y} \tag{12}$$

and

$$\text{tr}(\tilde{\underset{\sim}{V}}^{-1}\underset{\sim}{Z}_i\underset{\sim}{Z}_i') = (\underset{\sim}{y} - \underset{\sim}{X}\tilde{\underset{\sim}{\alpha}})'\tilde{\underset{\sim}{V}}^{-1}\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\tilde{\underset{\sim}{V}}^{-1}(\underset{\sim}{y} - \underset{\sim}{X}\tilde{\underset{\sim}{\alpha}}) \tag{13}$$

$$\text{for } i = 0, 1, \cdots, c.$$

Solutions restricted by $\tilde{\sigma}_0^2 > 0$ and $\tilde{\sigma}_i^2 \geq 0$ for $i = 1, \cdots, c$ are ML estimators.

Equivalent equations, given by Hartley and Rao [1967] are

$$\underset{\sim}{X}'\tilde{\underset{\sim}{H}}^{-1}\underset{\sim}{X}\tilde{\underset{\sim}{\alpha}} = \underset{\sim}{X}'\tilde{\underset{\sim}{H}}^{-1}\underset{\sim}{y} \tag{14}$$

$$\tilde{\sigma}_0^2 = (\underset{\sim}{y} - \underset{\sim}{X}\tilde{\underset{\sim}{\alpha}})'\tilde{\underset{\sim}{H}}^{-1}(\underset{\sim}{y} - \underset{\sim}{X}\tilde{\underset{\sim}{\alpha}})/N \tag{15}$$

$$\text{tr}(\tilde{\underset{\sim}{H}}^{-1}\underset{\sim}{Z}_i\underset{\sim}{Z}_i') = (\underset{\sim}{y} - \underset{\sim}{X}\tilde{\underset{\sim}{\alpha}})'\tilde{\underset{\sim}{H}}^{-1}\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\tilde{\underset{\sim}{H}}^{-1}(\underset{\sim}{y} - \underset{\sim}{X}\tilde{\underset{\sim}{\alpha}})/\tilde{\sigma}_0^2 \tag{16}$$

$$\text{for } i = 1, 2, \cdots, c.$$

An alternative form of (13) is

$$\{\text{tr}(\tilde{\underset{\sim}{V}}^{-1}\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\tilde{\underset{\sim}{V}}^{-1}\underset{\sim}{Z}_j\underset{\sim}{Z}_j')\}\tilde{\underset{\sim}{\sigma}}^2 = \{\underset{\sim}{y}'\tilde{\underset{\sim}{P}}\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\tilde{\underset{\sim}{P}}\underset{\sim}{y}\} \tag{17}$$

$$\text{for } i,j = 0, 1, \cdots, c.$$

4. REML: Restricted Maximum Likelihood

4.1. Rationale

Assume normality, as with ML, but maximize only that portion of the log likelihood which is invariant to $E(y)$. (See Thompson [1962], Patterson and Thompson [1971], and Corbeil and Searle [1976a, b].) This is equivalent to maximizing

$$L_1 = \text{constant} - \tfrac{1}{2}\log|\underset{\sim}{V}| - \tfrac{1}{2}\log|\underset{\sim}{X}^{*\prime}\underset{\sim}{V}^{-1}\underset{\sim}{X}^{*}| - \tfrac{1}{2}\underset{\sim}{y}'\underset{\sim}{P}\underset{\sim}{y} \tag{18}$$

for $\underset{\sim}{X}^{*}$ being any p^{*} (rank of $\underset{\sim}{X}$) linearly independent columns of $\underset{\sim}{X}$.

4.2. Method

Maximizing L_1 with respect to the σ^2's implicit in $\underset{\sim}{V}$ (and hence in $\underset{\sim}{P}$, also) leads to equations

$$\text{tr}(\hat{\underset{\sim}{P}}\underset{\sim}{Z}_i\underset{\sim}{Z}_i') = \underset{\sim}{y}'\hat{\underset{\sim}{P}}\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\hat{\underset{\sim}{P}}\underset{\sim}{y} \quad \text{for } i = 0, \cdots, c \tag{19}$$

equivalent to

$$\{\mathrm{tr}(\hat{P}Z_iZ_i'\hat{P}Z_jZ_j')\}\hat{\sigma}^2 = \{y'\hat{P}Z_iZ_i'\hat{P}y\} \qquad (20)$$

for i,j = 0, 1, ···, c.

With $\hat{\hat{\alpha}}$ defined by $X'\hat{V}^{-1}X\hat{\hat{\alpha}} = X'\hat{V}^{-1}y$, equation (19) also leads to

$$\hat{\sigma}_0^2 = (y - X\hat{\hat{\alpha}})'\hat{H}^{-1}(y - X\hat{\hat{\alpha}})/(N - p^*). \qquad (21)$$

Solutions for $\hat{\sigma}_0^2$ and $\hat{\sigma}^2$, restricted to $\hat{\sigma}_0^2 > 0$ and $\hat{\sigma}_i^2 \geq 0$ for i = 1, ···, c are REML estimators.

4.3. An equivalent procedure

Maximize the log likelihood of $K'y$ where K' is any matrix of full row rank $N - p^*$ such that $K'X = 0$:

$$L(K'y) = -\tfrac{1}{2}(N - p^*)\log 2\pi - \tfrac{1}{2}\log|K'VK| - \tfrac{1}{2}y'K(K'VK)^{-1}K'y. \qquad (22)$$

Comments

(i) Elements of $K'y$ are called "error contrasts", Harville [1977].

(ii) $K(K'VK)^{-1}K' = P$.

(iii) $L(K'y) - L_1$ = constant not dependent on α or $\dot{\sigma}^2$.

(iv) Corbeil and Searle [1976a] use a special form of K': define the fixed effects part of the model as the vector or cell means of those submost cells of the fixed effects factors that contain data, k such cells, say, with n_1, ···, n_k observations. Then delete the n_1^{th}, $(n_1 + n_2)^{th}$, ···, $(n_1 + n_2 + \cdots + n_k)^{th}$ rows from $I_N - \mathrm{diag}\{J_{n_1}, \cdots, J_{n_k}\}$, and the matrix remaining is a K'. It is called T by Searle and Corbeil.

5. MINQUE: Minimum Norm Quadratic Unbiased Estimation

5.1. Rationale

Estimate $p'\dot{\sigma}^2$, a linear function of the σ^2's by a quadratic form $y'Ay$, choosing A so that the estimator $y'Ay$

(i) is translation invariant: $\alpha \to \alpha + \delta$ does not alter $y'Ay$, $\Rightarrow AX = 0$.

(ii) is unbiased: $E(y'Ay) = p'\sigma^2$.

(iii) minimizes the weighted norm:

$$\|\dot{D}_w^{\frac{1}{2}}(\dot{Z}'A\dot{Z} - \dot{\Delta})\dot{D}_w^{\frac{1}{2}}\| \qquad (23)$$

where, for arbitrary weights w_i and for $r_i \equiv p_i/q_i$ i = 0, 1, ···, c,

$$\dot{D}_w = \mathrm{diag}\{w_0 I_{q_0} \cdots w_c I_{q_c}\}$$

and

$$\dot{\Delta} = \mathrm{diag}\{r_0 I_{q_0} \cdots r_c I_{q_c}\}. \qquad (24)$$

Were the elements of b actually known, a "natural estimator" of $p'\dot{\sigma}^2$ would be (Rao [1972])

$$\sum_{i=0}^{c} p_i \sum_{j=1}^{q_i} b_{ij}^2/q_i = b'\dot{\Delta}b .$$

5.2. Method

Minimizing (23) can be shown equivalent to

$$\text{minimizing } \mathrm{tr}(AV_w)^2 \qquad (25)$$

and this leads to solving the equations

$$\{\mathrm{tr}(P_wZ_iZ_i'P_wZ_jZ_j')\}\hat{\sigma}^2 = \{y'P_wZ_iZ_i'P_wy\} \qquad (26)$$

for i,j = 0, ···, c,

where

$$V_w = ZD_wZ' \quad \text{and} \quad P_w = V_w^{-1} - V_w^{-1}X(X'V_w^{-1}X)^{-}X'V_w^{-1} . \qquad (27)$$

Comments

(i) No distributional properties assumed.

(ii) No iteration: for given w, solve (26).

(iii) Special cases

(a) $w_i = 1$, for all i: $V_w = \sum_{i=0}^{c} Z_iZ_i'$.

(b) $w_0 = 1$, $w_i = 1$ for $i \geq 1$: $V_w = I$,

$P_w = I - X(X'X)^{-}X'$.

6. I-MINQUE: Iterative MINQUE

6.1. Rationale

The occurrence of w_i in V_w of (27) is exactly the same as that of σ_i^2 in V of (9). This prompts the idea of using the MINQUE equations iteratively — using $\hat{\sigma}^2$ as the w for a succeeding set of equations, which are then solved for $\hat{\sigma}^2$ — and so on.

6.2. Method

Solve, iteratively,

$$\{\mathrm{tr}(\hat{P}Z_iZ_i'\hat{P}Z_jZ_j')\}\hat{\sigma}^2 = \{y'\hat{P}Z_iZ_i'\hat{P}y\} \qquad (28)$$

for i,j = 0, 1, ···, c.

7. MIVQUE: Minimum Variance Quadratic Unbiased Estimation

7.1. Rationale

The same as MINQUE, except that instead of minimizing a norm, one minimizes the variance of the estimator $\underset{\sim}{y}'\underset{\sim}{A}\underset{\sim}{y}$. Combined with translation invariance and unbiasedness this involves minimizing

$$v(\underset{\sim}{y}'\underset{\sim}{A}\underset{\sim}{y}) = 2\mathrm{tr}(\underset{\sim}{A}\underset{\sim}{V})^2 + \sum_{i=0}^{c} \gamma_i \sigma_i^4 \lambda_i \qquad (29)$$

for

λ_i = sum of squares of diagonal elements of $\underset{\sim}{Z}_i'\underset{\sim}{A}\underset{\sim}{Z}_i$,

and where γ_i is the kurtosis parameter for $\underset{\sim}{b}_i$ (see, e.g., Anderson et al. [1977]). Under normality this is equivalent to

$$\text{minimizing } \mathrm{tr}(\underset{\sim}{A}\underset{\sim}{V})^2 \qquad (30)$$

similar to (25).

7.2. Method

Solve the equations

$$\{\mathrm{tr}(\hat{\underset{\sim}{P}}\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\hat{\underset{\sim}{P}}\underset{\sim}{Z}_j\underset{\sim}{Z}_j')\}\hat{\underset{\sim}{\sigma}}^2 = \{\underset{\sim}{y}'\hat{\underset{\sim}{P}}\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\hat{\underset{\sim}{P}}\underset{\sim}{y}\} \qquad (31)$$

$$\text{for } i = 0, \cdots, c.$$

8. MME: Mixed Model Equations

In the mixed linear model (1), equations designed for estimating the fixed effects $\underset{\sim}{\alpha}$ and for predicting the random effects $\underset{\sim}{b}$ also have uses in calculating estimates of variance components. They have computational uses, but provide no new estimation procedure.

The generalized least squares (Aitkin) equations for $\underset{\sim}{\alpha}$ are $\underset{\sim}{X}'\underset{\sim}{V}^{-1}\underset{\sim}{X}\hat{\underset{\sim}{\alpha}} = \underset{\sim}{X}'\underset{\sim}{V}^{-1}\underset{\sim}{y}$. If $\underset{\sim}{b}$, instead of representing random effects, were to represent fixed effects the normal equations for $\underset{\sim}{\alpha}$ and that $\underset{\sim}{b}$ would be

$$\begin{bmatrix} \underset{\sim}{X}'\underset{\sim}{R}^{-1}\underset{\sim}{X} & \underset{\sim}{X}'\underset{\sim}{R}^{-1}\underset{\sim}{Z} \\ \underset{\sim}{Z}'\underset{\sim}{R}^{-1}\underset{\sim}{X} & \underset{\sim}{Z}'\underset{\sim}{R}^{-1}\underset{\sim}{Z} \end{bmatrix} \begin{bmatrix} \underset{\sim}{\alpha}^0 \\ \underset{\sim}{b}^0 \end{bmatrix} = \begin{bmatrix} \underset{\sim}{X}'\underset{\sim}{R}^{-1}\underset{\sim}{y} \\ \underset{\sim}{Z}'\underset{\sim}{R}^{-1}\underset{\sim}{y} \end{bmatrix}. \qquad (32)$$

The MME's developed by Henderson et al. [1959] and Henderson [1963] are (32) adapted by adding $\underset{\sim}{D}^{-1}$ to $\underset{\sim}{Z}'\underset{\sim}{R}^{-1}\underset{\sim}{Z}$:

$$\begin{bmatrix} \underset{\sim}{X}'\underset{\sim}{R}^{-1}\underset{\sim}{X} & \underset{\sim}{X}'\underset{\sim}{R}^{-1}\underset{\sim}{Z} \\ \underset{\sim}{Z}'\underset{\sim}{R}^{-1}\underset{\sim}{X} & \underset{\sim}{Z}'\underset{\sim}{R}^{-1}\underset{\sim}{Z} + \underset{\sim}{D}^{-1} \end{bmatrix} \begin{bmatrix} \tilde{\underset{\sim}{\alpha}} \\ \tilde{\underset{\sim}{b}} \end{bmatrix} = \begin{bmatrix} \underset{\sim}{X}'\underset{\sim}{R}^{-1}\underset{\sim}{y} \\ \underset{\sim}{Z}'\underset{\sim}{R}^{-1}\underset{\sim}{y} \end{bmatrix}. \qquad (33)$$

These have also been written by Harville [1977] as

$$\begin{bmatrix} \underset{\sim}{X}'\underset{\sim}{R}^{-1}\underset{\sim}{X} & \underset{\sim}{X}'\underset{\sim}{R}^{-1}\underset{\sim}{Z}\underset{\sim}{D} \\ \underset{\sim}{Z}'\underset{\sim}{R}^{-1}\underset{\sim}{X} & \underset{\sim}{Z}'\underset{\sim}{R}^{-1}\underset{\sim}{Z}\underset{\sim}{D} + \underset{\sim}{I} \end{bmatrix} \begin{bmatrix} \tilde{\underset{\sim}{\alpha}} \\ \tilde{\underset{\sim}{\nu}} \end{bmatrix} = \begin{bmatrix} \underset{\sim}{X}'\underset{\sim}{R}^{-1}\underset{\sim}{y} \\ \underset{\sim}{Z}'\underset{\sim}{R}^{-1}\underset{\sim}{y} \end{bmatrix} \qquad (34)$$

with $\tilde{\underset{\sim}{b}} = \underset{\sim}{D}\tilde{\underset{\sim}{\nu}}$.

Comments

(i) $\tilde{\underset{\sim}{\alpha}}$ of (33) and (34) equals the Aitkin $\hat{\underset{\sim}{\alpha}}$.

(ii) $\tilde{\underset{\sim}{b}}$ of (33) is the BLUP (best linear unbiased predictor) of $\underset{\sim}{b}$.

(iii) $\tilde{\underset{\sim}{b}} = \underset{\sim}{D}\underset{\sim}{Z}'\underset{\sim}{V}^{-1}(\underset{\sim}{y} - \underset{\sim}{X}\tilde{\underset{\sim}{\alpha}})$, and with normality assumptions this is the ML estimator of $E(\underset{\sim}{b}|\underset{\sim}{y})$. In genetics $\tilde{\underset{\sim}{b}}$ is used for estimating genetic values.

(iv) In equations (33) the partitioned matrix on the l.h.s. is symmetric and p.s.d. whereas that in (34) is not.

(v) Equations (34) can be used with non-singular $\underset{\sim}{D}$ whereas (33) cannot.

(vi) Define

$$\underset{\sim}{T}^* = (\underset{\sim}{Z}'\underset{\sim}{R}^{-1}\underset{\sim}{Z}\underset{\sim}{D} + \underset{\sim}{I})^{-1} = (\underset{\sim}{Z}'\underset{\sim}{Z}\underset{\sim}{D}/\sigma^2 + \underset{\sim}{I})^{-}$$

$$= \{\underset{\sim}{T}^*_{ij}\} \quad \text{for } i,j = 1, \cdots, c. \qquad (35)$$

(vii) Define the lower right sub-matrix of the generalized inverse of the partitioned matrix on the l.h.s. of (34) as $\underset{\sim}{T}^{-1}$. Then

$$\underset{\sim}{T} = [\underset{\sim}{I} + \underset{\sim}{Z}'\{\underset{\sim}{R}^{-1} - \underset{\sim}{R}^{-1}\underset{\sim}{X}(\underset{\sim}{X}'\underset{\sim}{R}^{-1}\underset{\sim}{X})^{-}\underset{\sim}{X}'\underset{\sim}{R}^{-1}\}\underset{\sim}{Z}\underset{\sim}{D}]^{-1}$$

$$= \{\underset{\sim}{T}_{ij}\} \quad \text{for } i,j = 1, \cdots, c. \qquad (36)$$

(viii) $\underset{\sim}{T}^*$ and $\underset{\sim}{T}$ can be used in computing ML, REML, and I-MINQUE estimators of $\dot{\underset{\sim}{\sigma}}^2$.

9. THE DISPERSION-MEAN MODEL

9.1. Rationale

Estimate σ_i^2 unbiasedly by $\underset{\sim}{y}'\underset{\sim}{K}\underset{\sim}{A}\underset{\sim}{K}'\underset{\sim}{y}$ for $\underset{\sim}{K}'\underset{\sim}{y}$ being error contrasts. This leads to the model (see Pukelsheim [1976])

$$E(\underset{\sim}{y}) = \underset{\sim}{\mathcal{X}}\underset{\sim}{\sigma}^2 \quad \text{for} \quad \underset{\sim}{y} = \underset{\sim}{K}'\underset{\sim}{y} * \underset{\sim}{K}'\underset{\sim}{y} \qquad (37)$$

and

$$\underset{\sim}{\mathcal{X}} = (\underset{\sim}{K}' * \underset{\sim}{K}')\underset{\sim}{B} \quad \text{with} \quad \underset{\sim}{B} = [\mathrm{vec}(\underset{\sim}{Z}_0\underset{\sim}{Z}_0') \cdots \mathrm{vec}(\underset{\sim}{Z}_c\underset{\sim}{Z}_c')].$$

Also,

$$\underset{\sim}{W} = \text{var}(\underset{\sim}{y}) = (\underset{\sim}{K}' * \underset{\sim}{K}')\underset{\sim}{F}(\underset{\sim}{K} * \underset{\sim}{K})$$

for

$$\underset{\sim}{F} = (\underset{\sim}{V} * \underset{\sim}{V})(\underset{\sim}{I}_{N^2} + \underset{\sim}{I}_{(N,N)}) + (\dot{\underset{\sim}{Z}} * \dot{\underset{\sim}{Z}})\left[\text{diag}\{\text{vec}[\text{diag}(\gamma_0\sigma_0^4\underset{\sim}{I}_{q_0}\cdots\gamma_c\sigma_c^4\underset{\sim}{I}_{q_c})]\}\right](\dot{\underset{\sim}{Z}}' * \dot{\underset{\sim}{Z}}') \quad (38)$$

(see, for example, Anderson et al. [1977]).

9.2. Methods

(a) Ordinary least squares estimation is

$$\underset{\sim}{X}'\underset{\sim}{X}\hat{\underset{\sim}{\sigma}}^2 = \underset{\sim}{X}'\underset{\sim}{y}$$

and this becomes

$$\{\text{tr}(\underset{\sim}{K}\underset{\sim}{K}'\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\underset{\sim}{K}\underset{\sim}{K}'\underset{\sim}{Z}_j\underset{\sim}{Z}_j')\}\hat{\underset{\sim}{\sigma}}^2 = \{\underset{\sim}{y}'\underset{\sim}{K}\underset{\sim}{K}'\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\underset{\sim}{K}\underset{\sim}{K}'\underset{\sim}{y}\} \quad (39)$$

$$\text{for } i,j = 0, \cdots, c.$$

(b) "Generalized least squares" estimation is

$$\underset{\sim}{X}'\underset{\sim}{W}^-\underset{\sim}{X}\hat{\underset{\sim}{\sigma}}^2 = \underset{\sim}{X}'\underset{\sim}{W}^-\underset{\sim}{y} \; . \quad (40)$$

Under normality, $\gamma_i = 0$, which reduces $\underset{\sim}{W}$ to

$$\underset{\sim}{W} = (\underset{\sim}{K}'\underset{\sim}{V}\underset{\sim}{K} * \underset{\sim}{K}'\underset{\sim}{V}\underset{\sim}{K})(\underset{\sim}{I}_{m^2} + \underset{\sim}{I}_{(m,m)})$$

with $m \equiv N - p^*$ and

$$\underset{\sim}{W}^- = \tfrac{1}{4}(\underset{\sim}{I}_{m^2} + \underset{\sim}{I}_{(m,m)})[(\underset{\sim}{K}'\underset{\sim}{V}\underset{\sim}{K})^{-1} * (\underset{\sim}{K}'\underset{\sim}{V}\underset{\sim}{K})^{-1}]. \quad (41)$$

10. RELATIONSHIPS AMONG THE METHODS

10.1. REML and ANOVA, for balanced data

In most, if not all, cases of balanced data, solutions to the REML equations are identical to ANOVA estimators (obtained by equating analysis of variance sums of squares to their expected values).

10.2. ML and REML

ML: $$\{\text{tr}(\tilde{\underset{\sim}{V}}^{-1}\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\tilde{\underset{\sim}{V}}^{-1}\underset{\sim}{Z}_j\underset{\sim}{Z}_j')\}\tilde{\underset{\sim}{\sigma}}^2 = \{\underset{\sim}{y}'\tilde{\underset{\sim}{P}}\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\tilde{\underset{\sim}{P}}\underset{\sim}{y}\} \quad (17)$$

REML: $$\{\text{tr}(\hat{\underset{\sim}{P}}\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\hat{\underset{\sim}{P}}\underset{\sim}{Z}_j\underset{\sim}{Z}_j')\}\hat{\underset{\sim}{\sigma}}^2 = \{\underset{\sim}{y}'\hat{\underset{\sim}{P}}\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\hat{\underset{\sim}{P}}\underset{\sim}{y}\} \quad (20)$$

Note: $\tilde{\underset{\sim}{V}}^{-1}$ in the l.h.s. for ML is replaced by $\hat{\underset{\sim}{P}}$ in REML.

ML: $$\tilde{\sigma}_0^2 = (\underset{\sim}{y} - \underset{\sim}{X}\tilde{\underset{\sim}{\alpha}})'\tilde{\underset{\sim}{H}}^{-1}(\underset{\sim}{y} - \underset{\sim}{X}\tilde{\underset{\sim}{\alpha}})/N \quad (15)$$

REML: $$\hat{\sigma}_0^2 = (\underset{\sim}{y} - \underset{\sim}{X}\hat{\tilde{\underset{\sim}{\alpha}}})'\hat{\underset{\sim}{H}}^{-1}(\underset{\sim}{y} - \underset{\sim}{X}\hat{\tilde{\underset{\sim}{\alpha}}})/(N - p^*) \quad (21)$$

Note: Degrees of freedom for fixed effects are taken account of in REML.

10.3. REML and MINQUE

Choosing an initial arbitrary value for $\dot{\underset{\sim}{\sigma}}^2$ in $\dot{\underset{\sim}{P}}$ of (20), makes (20) the same as (26) when that same arbitrary value is used for $\underset{\sim}{w}$ in (26). Hence any first iterate of REML is a MINQUE.

10.4. REML, I-MINQUE, and MIVQUE under normality

Since (20), (28) and (31) are the same equations, REML, I-MINQUE and MIVQUE under normality estimators are the same.

10.5. ML and MME

Using $\tilde{\underset{\sim}{\alpha}}$ and $\tilde{\underset{\sim}{b}}$ from the MME (33)

$$\tilde{\sigma}_0^2 = \underset{\sim}{y}'(\underset{\sim}{y} - \underset{\sim}{X}\tilde{\underset{\sim}{\alpha}} - \underset{\sim}{Z}\tilde{\underset{\sim}{b}})/N$$

and with $\underset{\sim}{T}^*$ of (35)

$$\tilde{\sigma}_i^{2(r+1)} = \frac{(\tilde{\underset{\sim}{b}}_i'\tilde{\underset{\sim}{b}}_i)^{(r)} + \tilde{\sigma}_i^{2(r)}\text{tr}(\underset{\sim}{T}_{ii}^{*(r)})}{q_i} \; ;$$

$$\tilde{\sigma}_i^{2(r+1)} = \frac{(\tilde{\underset{\sim}{b}}_i'\tilde{\underset{\sim}{b}}_i)^{(r)}}{q_i - \text{tr}(\underset{\sim}{T}_{ii}^{*(r)})} \; . \quad (42)$$

Note: The second expression in (42) always yields positive estimates.

10.6. REML and MME

$$\hat{\sigma}_0^2 = \underset{\sim}{y}'(\underset{\sim}{y} - \underset{\sim}{X}\tilde{\underset{\sim}{\alpha}} - \underset{\sim}{Z}\tilde{\underset{\sim}{b}})/(N - p^*)$$

and

$$\hat{\sigma}_i^{2(r+1)} = \frac{(\tilde{\underset{\sim}{b}}_i'\tilde{\underset{\sim}{b}}_i)^{(r)} + \hat{\sigma}_i^{2(r)}\text{tr}(\underset{\sim}{T}_{ii}^{(r)})}{q_i} \; ;$$

$$\hat{\sigma}_i^{2(r+1)} = \frac{(\tilde{\underset{\sim}{b}}_i'\tilde{\underset{\sim}{b}}_i)^{(r)}}{q_i - \text{tr}(\underset{\sim}{T}_{ii}^{(r)})} \; . \quad (43)$$

Note: The second expression in (43) always yields positive estimates. Equations (43) are the same as (42) with $\underset{\sim}{T}_{ii}$ in place of $\underset{\sim}{T}_{ii}^*$.

10.7. MINQUE and MME

Equations (20), (26), (28) and (31), for REML, MINQUE, I-MINQUE and MIVQUE under normality, respectively, are essentially of the form

$$\{\text{tr}(\underset{\sim}{P}\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\underset{\sim}{P}\underset{\sim}{Z}_j\underset{\sim}{Z}_j')\}\hat{\underset{\sim}{\sigma}}^2 = \{\underset{\sim}{y}'\underset{\sim}{P}\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\underset{\sim}{P}\underset{\sim}{y}\} \quad (44)$$

$$\text{for } i,j = 0, 1, \cdots, c.$$

These are equivalent to

$$\begin{bmatrix} \mathrm{tr}(\underset{\sim}{P}^2) & \{\mathrm{tr}(\underset{\sim}{P}^2\underset{\sim}{Z}_j\underset{\sim}{Z}_j')\} \\ \{\mathrm{tr}(\underset{\sim}{P}^2\underset{\sim}{Z}_i\underset{\sim}{Z}_i')\} & \{\mathrm{tr}(\underset{\sim}{P}\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\underset{\sim}{P}\underset{\sim}{Z}_j\underset{\sim}{Z}_j')\} \end{bmatrix} \begin{bmatrix} \hat{\sigma}_0^2 \\ \hat{\underset{\sim}{\sigma}}^2 \end{bmatrix} = \begin{bmatrix} \underset{\sim}{y}'\underset{\sim}{P}^2\underset{\sim}{y} \\ \underset{\sim}{y}'\underset{\sim}{P}\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\underset{\sim}{P}\underset{\sim}{y} \end{bmatrix}$$

$$\text{for } i,j = 1, \cdots, c.$$

The terms in this equation can be expressed as functions of sub-matrices of $\underset{\sim}{T}$ of (36) as follows:

$$\mathrm{tr}(\underset{\sim}{P}^2) = \frac{N - r}{\sigma_0^4} + \frac{1}{\sigma_0^4}\sum_{i=1}^{c}\sum_{j=1}^{c}\mathrm{tr}(\underset{\sim}{T}_{ij}\underset{\sim}{T}_{ji})$$

$$\mathrm{tr}(\underset{\sim}{P}^2\underset{\sim}{Z}_i\underset{\sim}{Z}_i') = \frac{1}{\sigma_0^2\sigma_i^2}\left[\mathrm{tr}(\underset{\sim}{T}_{ii}) - \sum_{j=1}^{c}\mathrm{tr}(\underset{\sim}{T}_{ij}\underset{\sim}{T}_{ji})\right]$$

$$\text{for } i = 1, \cdots, c$$

$$\mathrm{tr}(\underset{\sim}{P}\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\underset{\sim}{P}\underset{\sim}{Z}_i\underset{\sim}{Z}_i') = \frac{q_i}{\sigma_0^4} - \frac{2}{\sigma_i^4}\mathrm{tr}(\underset{\sim}{T}_{ii}) + \frac{1}{\sigma_i^4}\mathrm{tr}(\underset{\sim}{T}_{ii})^2$$

$$\text{for } i = 1, \cdots, c$$

$$\mathrm{tr}(\underset{\sim}{P}\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\underset{\sim}{P}\underset{\sim}{Z}_j\underset{\sim}{Z}_j') = \frac{1}{\sigma_i^2\sigma_j^2}\mathrm{tr}(\underset{\sim}{T}_{ij}\underset{\sim}{T}_{ji}) \text{ for } i \neq j = 1, \cdots, c$$

$$\underset{\sim}{y}'\underset{\sim}{P}^2\underset{\sim}{y} = (\underset{\sim}{y}'\underset{\sim}{y} - \tilde{\underset{\sim}{\alpha}}'\underset{\sim}{X}'\underset{\sim}{y} - \tilde{\underset{\sim}{b}}'\underset{\sim}{Z}'\underset{\sim}{y})/\sigma_0^4 - \frac{1}{\sigma_0^2}\sum_{i=1}^{c}\frac{\tilde{\underset{\sim}{b}}_i'\tilde{\underset{\sim}{b}}_i}{\sigma_i^2}$$

$$\underset{\sim}{y}'\underset{\sim}{P}\underset{\sim}{Z}_i\underset{\sim}{Z}_i'\underset{\sim}{P}\underset{\sim}{y} = \tilde{\underset{\sim}{b}}_i'\tilde{\underset{\sim}{b}}_i/\sigma_i^4 \text{ for } i = 1, \cdots, c.$$

10.8. Dispersion-Mean model and MINQUE

The "generalized least squares" equations (40) of the dispersion-mean model, when assuming normality and using $\underset{\sim}{W}$ of (41), reduce to (44). When an assigned value, $\underset{\sim}{w}$ say, is used for $\hat{\underset{\sim}{\sigma}}^2$ in $\underset{\sim}{P}$, (44) is the same as (26) for MINQUE; if equations (44) are solved iteratively, then they are equivalent to (20), (28) and (31) for REML, I-MINQUE, and MIVQUE under normality, respectively.

REFERENCES

(1) Anderson, R. L. and Bancroft, T. A. [1952]. Statistical Theory in Research, McGraw-Hill, New York.

(2) Anderson, R. D., Quaas, R. L. and Searle, S. R. [1977]. Fourth moments in the general linear model; and the variance of translation invariant quadratic forms. Paper No. BU-630-M, Biometrics Unit, Cornell University.

(3) Corbeil, R. R. and Searle, S. R. [1976a]. Restricted maximum likelihood (REML) estimation of variance components in the mixed model. Technometrics 18:31-38.

(4) Corbeil, R. R. and Searle, S. R. [1976b]. A comparison of variance components estimators. Biometrics 32:779-791.

(5) Daniels, H. E. [1939]. The estimation of components of variance. J. Roy. Stat. Soc. Supp. 6:186-197.

(6) Graybill, F. A. and Hultquist, R. A. [1961]. Theorems concerning Eisenhart's Model II. Ann. Math. Stat. 32:261-269.

(7) Graybill, F. A. and Wortham, A. W. [1956]. A note on uniformly best unbiased estimators for variance components. J. Am. Stat. Assoc. 51:266-268.

(8) Hartley, H. O. and Rao, J. N. K. [1967]. Maximum likelihood estimation for the mixed analysis of variance model. Biometrika 54:93-108.

(9) Harville, D. A. [1977]. Maximum likelihood approaches to variance component estimation and to related problems. J. Am. Stat. Assoc. 72:320-340.

(10) Henderson, C. R. [1953]. Estimation of variance and covariance components. Biometrics 9: 226-252.

(11) Henderson, C. R. [1963]. Selection index and expected genetic advance. Statistical Genetics and Plant Breeding, National Academy of Sciences – National Research Council Publication No. 982, 141-163.

(12) Henderson, C. R., Kempthorne, O., Searle, S. R. and von Krosigk, C. N. [1959]. Estimation of environmental and genetic trends from records subject to culling. Biometrics 15: 192-218.

(13) Patterson, H. D. and Thompson, R. [1971]. Recovery of interblock information when block sizes are unequal. Biometrika 58:545-554.

(14) Pukelsheim, F. [1976]. Estimating variance components in linear models. J. Multivar. Anal. 6:626-629.

(15) Rao, C. R. [1972]. Estimation of variance and covariance components in linear models. J. Am. Stat. Assoc. 67:112-115.

(16) Searle, S. R. and Quaas, R. L. [1978]. A detailed description of recent methods of estimating variance components, with applications in animal breeding. Mimeo Report (in preparation), Biometrics Unit, Cornell University.

(17) Thompson, W. A. [1962]. The problem of negative estimates of variance components. Ann. Math. Stat. 33:273-289.

(18) Winsor, C. P. and Clarke, G. L. [1940]. A statistical study of variation in the catch of plankton nets. Sears Foundation J. Marine Res. 3:1-34.

WORKSHOP 3

COMPUTING FOR ECONOMETRICS

Chair: Warren Dent, University of Iowa

COMPUTATION OF THE EXACT LIKELIHOOD FOR AN ARMA PROCESS

Craig F. Ansley
University of Chicago

ABSTRACT

The likelihood function for an ARMA process is derived in a form which allows more efficient computation than other exact methods proposed in the literature.

Exact maximum likelihood estimators for the parameters of an ARMA process are often better than least squares estimators, particularly for a seasonal process. Moreover, a sum-of-squares approximation to the exact likelihood cannot be used for a noninvertible moving average process. Even in the calculation of the sum-of-squares function there can be loss of accuracy or numerical convergence problems close to the boundaries of stationarity or invertibility.

The algorithm described here suffers no loss of accuracy near a boundary, and can be used for both invertible and noninvertible moving average representations. The nature of the likelihood surface over the entire parameter space is discussed in this regard.

1. Introduction

Suppose that, using the notation of Box and Jenkins [2], the stationary process $\{w_t\}$ is generated by the autoregressive-moving average scheme of order (p, q):

$$w_t - \phi_1 w_{t-1} - \cdots - \phi_p w_{t-p} = a_t - \theta_1 a_{t-1} - \cdots - \theta_q a_{t-q}$$

i.e.,
$$\phi(B) w_t = \theta(B) a_t \qquad (1.1)$$

where B is the backshift operator. Given n observations $\underline{w}_n' = (w_1, \ldots, w_n)$ on the series, one seeks the maximum likelihood estimates of the parameters $\underline{\phi}' = (\phi_1, \ldots, \phi_p)$ and $\underline{\theta}' = (\theta_1, \ldots, \theta_q)$ in (1.1). The likelihood function for the observations is:

$$L(\underline{\phi}, \underline{\theta}, \sigma^2 | \underline{w}) = (2\pi\sigma^2)^{-n/2} |\Omega_n|^{-1/2} \exp\left\{ - \frac{\underline{w}_n' \Omega_n^{-1} \underline{w}_n}{2\sigma^2} \right\} \qquad (1.2)$$

where σ^2 is the variance of the random shocks a_t and $\sigma^2 \Omega_n$ is the $n \times n$ covariance matrix of $\underline{w}$.

In computing the likelihood (1.2), one encounters two problems: (i) calculation of the exponent $\underline{w}_n' \Omega_n^{-1} \underline{w}_n$ and (ii) calculation of the determinant $|\Omega_n|$. Because the term $|\Omega_n|$ is dominated by the term $\exp\{-\underline{w}_n' \Omega_n^{-1} \underline{w}_n / 2\sigma^2\}$ for large values of n, Box and Jenkins [2, p. 213] suggest disregarding the determinant for maximum likelihood estimation; this is often done in practice. The remaining expression can be formulated as a least squares estimation problem. However, it has often been suggested, for example by Kang [6] and Box, Hillmer and Tiao [1], that disregarding this determinant can lead to inferior estimators. Min [10] and Dent and Min [4] have a number of Monte Carlo findings supporting this view.

McLeod [7] has proposed a method whereby the determinant $|\Omega_n|$ is replaced by its asymptotic limit, and has conducted Monte Carlo experiments showing improvements in estimation for the cases he examined. His approach, however, does not avoid the problem of efficient computation of the exponent $\underline{w}_n' \Omega_n^{-1} \underline{w}_n$.

Two procedures for calculation of the exponent are suggested by Box and Jenkins. One is to approximate it by the sum of squares:

$$S = \sum_{t=p+1}^{n} \hat{a}_t^2$$

where $\hat{a}_t$ is calculated from the equations (1.1) under the assumption that the q starting values $a_p, \ldots, a_{p-q+1}$ are all zero. This is called the "conditional" method following its derivation from a conditional likelihood function. Box and Jenkins also show that in the unconditional likelihood (1.2):

$$\underline{w}_n' \Omega_n^{-1} \underline{w}_n = \sum_{-\infty}^{n} [a_t]^2$$

where $[a_t] = E[a_t | \underline{w}_n, \underline{\phi}, \underline{\theta}]$. They approximate the infinite sum by a sum from $-Q$ to n for a suitably large Q and obtain values of $[a_t]$ through their iterative "backforecasting" procedure [p. 213]. Newbold [12] points out that should a root of either the moving average or autoregressive operator lie close to the unit circle, it may be necessary to choose a very large Q and/or to iterate many times to reach convergence.

Pierce [14] has shown that the asymptotic properties of least squares are independent of the particular starting values shown and, therefore, that the conditional and backforecasting least squares estimators have the same properties for large n. However, for seasonal series Box and Jenkins [2, p. 211] state that "the conditional approximation is not very satisfactory and the unconditional calculation becomes even more necessary." Min [10], Dent and Min [4], and Nelson [11] have Monte Carlo results disputing this and showing that the conditional method often performs as well as, or better than, the backforecasting method, especially for pure moving average models, although the evidence for seasonal models is scant.

We present here a computational algorithm for calculating the exact likelihood that appears more efficient than other methods described in the literature. The speed of the algorithm depends mainly on the order of the moving average operator, the autoregressive order being unimportant except in short series. For pure autoregressive and pure moving average processes, refinements are possible that greatly improve efficiency. Moreover, the algorithm is exact and numerically stable outside the region of invertibility of the moving average operator.

2. Existing Procedures for Exact Likelihoods

Before describing our algorithm, we discuss briefly two existing procedures for calculating exact likelihoods.

Newbold [12] has derived the exact likelihood for a mixed process using a generalization of the approach used by Box and Jenkins [2] to obtain the exact likelihood for a pure moving average process. If we write $\underline{a}_n' = (a_1, \ldots, a_n)$ and $\underline{a}^{*\prime} = (a_{1-p-q}, \ldots, a_0)$ where $a_{1-p-j} = a_{1-j}$, $j = 1, \ldots, q$ and $a_{1-j} = w_{1-j}$, $j = 1, \ldots, p$, then we can write the equations (1.1) for $t = 1, \ldots, n$ as:

$$\begin{pmatrix} \underline{a}^* \\ \underline{a}_n \end{pmatrix} = \begin{pmatrix} 0 \\ L_n \end{pmatrix} \underline{w}_n + \begin{pmatrix} I \\ X_n \end{pmatrix} \underline{a}^* \ . \tag{2.1}$$

If the covariance matrix of $\underline{a}^*$ is Σ, we can find a matrix T such that $T \Sigma T' = I_r$ where $r = p + q$ and rewrite (2.1) as:

$$\begin{pmatrix} \underline{u}^* \\ \underline{u}_n \end{pmatrix} = \begin{pmatrix} 0 \\ L_n \end{pmatrix} \underline{w}_n + \begin{pmatrix} I_r \\ X_n T^{-1} \end{pmatrix} \underline{u}^*$$

i.e.,
$$\underline{u} = L\underline{w}_n + Z\underline{u}^* \tag{2.2}$$

where $\underline{u}^* = T\underline{a}^*$ and $\underline{u}_n = \underline{a}_n$, and the definitions of L and Z follow from the previous line. Newbold shows by a probabilistic argument that the exponent in (2.1) is given by:

$$\begin{aligned} \underline{w}_n'\Omega_n^{-1}\underline{w}_n &= L\underline{w}_n(I_{n+r} - Z(Z'Z)^{-1}Z)\underline{w}_n'L' \\ &= L_n\underline{w}_n(I_n - X_nT^{-1}(Z'Z)^{-1}(T^{-1})'X_n')\underline{w}_n'L_n' \ . \end{aligned} \tag{2.3}$$

This is a least-squares solution to the equations (2.2), which can be obtained efficiently by singular value decomposition of Z. (See, for example, Golub and Reinsch [5].) The determinant is given by $|\Omega_n| = |Z'Z|$.

Prothero [15] has developed a program for exact maximum likelihood estimation based on Newbold's approach.

Dent [3] independently developed an algorithm for exact maximum likelihood based on direct algebraic manipulation of $\underline{w}_n'\Omega_n\underline{w}_n$. Using the above notation where applicable, we can rewrite the equation (1.1) for $t = 1, \ldots, n$ in the form:

$$A_1\underline{w}_n - A_2\underline{a}_n - \Delta\underline{a}^* = 0 \ . \tag{2.4}$$

The exponent can be written as:

$$\underline{w}_n'\Omega_n^{-1}\underline{w}_n = A_2^{-1}A_1\underline{w}_nH^{-1}\underline{w}_n'A_1'(A_2^{-1})'$$

where

$$H = I_n + A_2^{-1}\Delta\ \Sigma\ \Delta'(A_2^{-1})' \ .$$

Multiplying (2.4) by A_2^{-1} we have:

$$\begin{pmatrix} \underline{a}^* \\ \underline{a}_n \end{pmatrix} = \begin{pmatrix} 0 \\ A_2^{-1}A_1 \end{pmatrix} \underline{w}_n - \begin{pmatrix} -I_r \\ A_2^{-1}\Delta \end{pmatrix} \underline{a}^*$$

whence $L_n = A_2^{-1}A_1$ and $X_n = -A_2^{-1}\Delta$, and

$$Z = -\begin{pmatrix} -I_r \\ A_2^{-1}\Delta T^{-1} \end{pmatrix} \ .$$

Thus

$$H = I_n + X_nT^{-1}(T^{-1})'X_n'$$

and

$$\begin{aligned} H^{-1} &= I_n - X_nT^{-1}(I_r + (T^{-1})'X_n'X_nT^{-1})(T^{-1})'X_n' \\ &= I_n - X_nT^{-1}(Z'Z)^{-1}(T^{-1})'X_n' \ , \end{aligned}$$

and by comparison with (2.3) we see that Dent's approach is identical to Newbold's. Dent goes on to suggest details of computation using singular value transformations and obtains a different expression $|\Omega_n| = |H|$ for the determinant, but carries out essentially the same operations as a singular value approach to Newbold's formulation.

3. A New Algorithm for Calculating the Likelihood

We now describe a simple method for calculating $\underline{w}_n'\Omega_n^{-1}\underline{w}_n$ and $|\Omega_n|$ in (1.2). Because Ω_n is positive definite, there exists a lower triangular matrix L_n such that $\Omega_n = L_nL_n'$. This is the Cholesky factorization. Writing $\underline{e} = L_n^{-1}\underline{w}_n$ we obtain:

$$\mathrm{Cov}(\underline{e}) = \sigma^2L_n^{-1}\Omega(L_n^{-1})' = \sigma^2I$$

so that $\underline{e}' = (e_1, \ldots, e_n)$ is a set of independent normal random variables with mean zero and variance σ^2. Further

$$\underline{w}_n'\Omega_n\underline{w}_n = \sum_1^n e_t^2 \tag{3.1}$$

and

$$|\Omega_n|^{-1/2} = |L_n|^{-1}$$

so that

$$p(\underline{w}_n|\underline{\phi}, \underline{\theta}, \sigma^2) = (2\pi\sigma^2)^{-n/2}|L_n|^{-1} \exp\left\{-\sum_1^n e_t^2/2\sigma^2\right\}$$

where $|L_n|$ is the product of its diagonal elements.

In general, the matrix Ω_n is $n \times n$, and direct evaluation of L_n and $\underline{e}_n$ would be infeasible. However, it is possible to use the structure of (1.1) to simplify the computations considerably.

Consider first the MA(q) process:

$$y_t = \theta(B)a_t .$$

The (ij)th element of Ω_n is given by:

$$\Omega_n(i, j) = \gamma_{|i-j|} \qquad |i - j| \leq q$$
$$= 0 \qquad |i - j| > q$$

where $\{\sigma^2\gamma_k\}$ is the autocovariance function of $\{w_t\}$. The matrix Ω_n is a symmetric positive definite band matrix with band width q; i.e., all the elements outside the qth co-diagonals are zero. The matrix L_n is thus also a band matrix with band width q.

A computationally efficient algorithm for decomposition of Ω_n to obtain L_n in this case has been published by Martin and Wilkinson [9]. The algorithm requires approximately $n(q + 1)(q + 2)/2$ multiplications and n square roots. In addition, the determinant $|L_n|$ can be obtained with n multiplications.

In general the matrix L_n^{-1} is not a band matrix. However, the linear equations $L_n\underline{e}_n = \underline{w}_n$ can be solved recursively with approximately $n(q + 1)$ multiplications, and L_n^{-1} need not be calculated explicitly.

Consider now the general ARMA(p, q) model (1.1). It is convenient to define a sequence $\underline{z}_n = (z_1, \ldots, z_n)$ by:

$$z_t = w_t \qquad t = 1, \ldots, m$$
$$= \phi(B)w_t \qquad t = m + 1, \ldots, n \qquad (3.2)$$

where $m = \max(p, q)$. It is clear that $\mathrm{Cov}(z_t, z_{t\pm s}) = 0$ for $s > m$. The covariance matrix $\mathrm{Cov}(\underline{z}) = \sigma^2\Omega_n^z$ is thus a positive definite symmetric band matrix, and we can use the Martin and Wilkinson algorithm to obtain the decomposition $\Omega_n^z = L_n^z L_n^{z\prime}$ and solve the equations $\underline{e}_n = (L_n^z)^{-1}\underline{z}_n$.

Because the Jacobian of the transformation (3.1) is 1, we have:

$$|\Omega_n|^{-1/2} = |L_n|^{-1} = |L_n^z|^{-1} .$$

The above approach is exact and efficient; a discussion of its computational properties is given below. It also has a number of other interesting properties, some of which are described in the following sections. In particular, because e_t is obtained by a linear combination of $w_1, \ldots, w_t$ for all t, this approach is equivalent to the Gram-Schmidt orthogonalization process except for the normalizing constant σ. It is thus apparent that for large n the e_t's will converge to the random shocks in the Wold representation, or, in the case $\theta(B)$ is invertible, to the random shocks a_t in (1).

4. Special Cases

We consider here three simplifications in special cases that greatly increase the computational efficiency of the algorithm.

1. Pure AR Processes

In the pure AR case, (3.1) reduces to:

$$z_t = w_t \qquad t = 1, \ldots, p$$
$$z_t = \phi(B)w_t \qquad t = p + 1, \ldots, n .$$

The matrix Ω_n^z has the form:

$$\Omega_n^z = \begin{bmatrix} \Omega_p & 0 \\ 0 & I \end{bmatrix}$$

and thus L_n^z has the form:

$$L_n^z = \begin{bmatrix} L_p & 0 \\ 0 & I \end{bmatrix}$$

where $\Omega_p = L_p L_p'$ and where $\sigma^2\Omega_p$ is the $p \times p$ covariance matrix of $z_1, \ldots, z_p$. In this case it is more efficient to obtain L_p by using the general decomposition algorithm of Martin, Peters and Wilkinson [8]. This algorithm uses p square roots and approximately $p^3/6$ multiplications. The values $e_1, \ldots, e_n$ can then be obtained from:

$$\underline{e}_p = L_p^{-1}\underline{z}_p$$
$$e_t = z_t \qquad t = p + 1, \ldots, n .$$

The equations $L_p\underline{e}_p = \underline{z}_p$ can be solved recursively, without inverting L_p explicitly.

It is interesting to note that a similar simplification can be made to the Newbold/Dent approach for pure AR processes also.

2. Seasonal MA Processes

The matrix Ω_n for the seasonal moving average process

$$y_t = \Theta(B^s)\theta(B)a_t \qquad (4.1)$$

consists of nonzero bands of constant width equidistant from the principal diagonal, with zeros elsewhere. The algorithm could be applied directly using a total band width of $q + sQ$, where s is the length of the seasonal cycle and Q the seasonal moving average order. However, it is possible to reduce the number of calculations by recognizing that the matrix L_n has the form:

$$L_n(i, j) = 0 \begin{cases} i = rs + k & r = 0, 1, 2, \ldots; \\ & k = 1, 2, \ldots, s - q \\ i - j = hs + \ell & h = 0, 1, 2, \ldots; \\ & \ell = q + 1, \ldots, s \\ i - j > q + sQ \\ j > i \end{cases}$$

providing $q < s/2$. To show this, we need the following lemma.

Lemma

Let $\{\gamma_k^{(1)}\}$ be the autocovariances of the MA(q) process:

$$w_t^{(1)} = \theta(B)a_t \qquad \sigma_a^2 = 1$$

and let $\{\gamma_k^{(2)}\}$ be the autocovariances of the MA(Q) process:

$$w_t^{(2)} = \Theta(B)a_t \qquad \sigma_a^2 = 1 .$$

Then if $q < s/2$, the autocovariances $\{\gamma_k\}$ of (4.1) are given by:

$$\gamma_{rs+k} = \gamma_r^{(2)}\,\gamma_k^{(1)} \qquad r = 0, 1, 2, \ldots;\ |k| \leq \tfrac{1}{2}s .$$

The proof follows immediately by noting that the generating function for the $\{\gamma_k\}$ is given by:

$$\gamma(u) = \Theta(u^s)\theta(u)\theta(u^{-1})\Theta(u^{-s}) = \gamma^{(1)}(u)\gamma^{(2)}(u^s) .$$

As noted above, the random variables $e_1, \ldots, e_n$ are obtained by the Gram-Schmidt orthogonalization of $w_1, \ldots, w_n$. It follows that the Hilbert space $H(w_1, \ldots, w_t)$ is spanned by $e_1, \ldots, e_t$. With this in mind, we can prove the following theorem, which is equivalent to our statement concerning the form of L_n.

Theorem

The Hilbert space $H(w_k, w_{k+s}, \ldots, w_{rs+k})$ generated by $w_k, w_{s+k}, \ldots, w_{rs+k}$ is a subspace of $H(e_{hs+k-j};\ h = 0, \ldots, r;\ j = 0, \ldots, q;\ hs + k - j \geq 1)$ provided $1 \leq k \leq s - q$.

Proof

We proceed by induction. The covariance matrix Ω_n is a symmetric band matrix with q subdiagonals up to and including row $s - q$. The corresponding Cholesky factorization is a lower triangular band matrix, with band width q, giving:

$$w_t = \sum_{j=0}^{\min(q,t-1)} c_j^{(t)} e_{t-j} \qquad t = 1, \ldots, s - q .$$

The result is thus established for $r = 0$.

Assume now the result holds for $r \leq m$. Write $w_{(m+1)s+k} = w^{(1)} + w^{(2)}$ where $w^{(1)}$ is the projection of $w_{(m+1)s+k}$ onto the subspace $H(w_k, w_{s+k}, \ldots, w_{ms+k})$ and $w^{(2)}$ is the corresponding perpendicular. We can write.

$$w^{(1)} = \sum_{j=0}^{m} b_j w_{js+k}$$

and

$$w^{(2)} = w_{(m+1)s+k} - \sum_{j=0}^{m} b_j w_{js+k} .$$

Taking covariances with w_{rs+k} for $0 \leq r \leq m$ we have, using the lemma,

$$\begin{aligned}\mathrm{Cov}(w^{(2)}, w_{rs+k}) &= \gamma_{(m+1-r)s} - \sum_{j=0}^{m} b_j \gamma_{(j-r)s} \\ &= \gamma_0^{(1)}\left[\gamma_{m+1-r}^{(2)} - \sum_{j=0}^{m} b_j \gamma_{j-r}^{(2)}\right] \\ &= 0\end{aligned}$$

because $w^{(2)}$ and w_{rs+k} are orthogonal. Thus:

$$\gamma_{m+1-r}^{(2)} - \sum_{j=0}^{m} b_j \gamma_{j-r}^{(2)} = 0 \qquad r = 0, \ldots, m . \quad (4.2)$$

We now show that $w^{(2)}$ is orthogonal to the space $H(w_j;\ j = 1, \ldots, ms + k + q^*)$ where $q^* = s - q - 1 \geq 0$. Choose a value of t in the range $1 \leq t \leq ms + k + q^*$ and write $t = rs + \ell;\ r = 0, \ldots, m;\ \ell = 1, \ldots, s - 1\ (r < m);$ $\ell = 1, \ldots, q^* + k\ (r = m)$.

Consider first the case $1 \leq \ell < k - q$.

$$\mathrm{Cov}(w^{(2)}, w_t) = \gamma_{(m+1-r)+k-\ell} - \sum_{j=0}^{m} b_j \gamma_{(j-r)s+k-\ell} = 0$$

because each covariance on the right-hand side is zero providing only that $k \leq s - q$.

A similar argument holds for the case $k + q < \ell \leq \min(q^*, s - 1)(r < m)$; $k + q \leq \ell \leq q^*\ (r = m)$, without restriction on k.

Consider now $k - q \leq \ell \leq k + q$, i.e., $|k - \ell| \leq q$.

$$\begin{aligned}\mathrm{Cov}(w^{(2)}, w_t) &= \gamma_{(m+1-r)s+k-\ell} - \sum_{j=0}^{m} b_j \gamma_{(j-r)s+k-\ell}^{(2)} \\ &= \gamma_{k-\ell}^{(1)}\left[\gamma_{(m+1-r)}^{(2)} - \sum_{j=0}^{m} b_j \gamma_{(j-r)}^{(2)}\right] \\ &= 0\end{aligned}$$

by (4.2)

Finally, consider the case $\ell = q^* + 1, \ldots, s - 1$, i.e., $\ell - k > s - q$. Note that this case is a possibility only if $r < m$.

$$\begin{aligned}\mathrm{Cov}(w^{(2)}, w_t) &= \gamma_{(m+1-r)+k-\ell} - \sum_{j=0}^{m} b_j \gamma_{(j-r-1)}^{(1)} \\ &= \gamma_{\ell-k+s}^{(1)}\left[\gamma_{m-r}^{(1)} - \sum_{j=0}^{m} b_j \gamma_{(j-r-1)}^{(1)}\right] \\ &= 0\end{aligned}$$

by (4.2) because $r < m$.

Thus $w^{(2)}$ is an element in the subspace $H(e_{(m+1)s+k-j};\ j = 0, \ldots, q)$. Now

$$w_{(m+1)s+k} = w^{(2)} + \sum_{j=0}^{m} b_j w_{js+k}$$

and by the induction hypothesis $w_{(m+1)s+k}$ is an element of $H(e_{hs+k-j};\ h = 0, \ldots, m + 1;$ $j = 0, \ldots, q;\ hs + k - j \geq 1)$. Q.E.D.

It follows from the proof that we can obtain the factors $b_j^{(m)}$ for the mth cycle by Cholesky factorization of the band matrix generated by the covariances $\{\gamma_{rs} = \gamma_0^{(l)}\gamma_k^{(r)}\}$. In this way, bands for successive cycles can be obtained from previously calculated bands, and only the factors $c_j^{(ms+k)}$, $j = 0, \ldots, q$ for cycle m calculated anew.

3. Mixed Processes

If we replace the transformation (3.2) by the transformation:

$$z_t = \phi(F)w_t \qquad t = 1, \ldots, n - \max(p, q)$$
$$z_t = w_t \qquad t = n - \max(p, q) + 1, \ldots, n \tag{4.3}$$

it is clear that the first $n-\max(p, q)$ rows of the matrix Ω_n^z will be identical to those of a moving average process with operator $\theta(B)$. If $q < p$, this means that all rows up the $n-\max(p, q)$ will have a Cholesky factorization for band width q, and the full band width need not be used except for the final p rows.

This has special significance for processes with a seasonal MA operator, because we can take advantage of the special form of L_n discussed above.

The matrix for our earlier formulation does not have this property. We note that the orthogonalization of the series z_t defined by (4.3) no longer corresponds to the Grăm-Schmidt orthogonalization.

5. Stationarity Conditions

We assumed in deriving the above algorithm that the process (1.1) was stationary. However, in some situations arising in maximum likelihood estimation, it is possible that an attempt may be made to evaluate the function (1.2) outside the region of stationarity. We examine some of the implications of this possibility in this section.

First, note that a pure moving average process is always stationary, and our assumption cannot be violated.

Consider now a pure autoregressive process. We assume the matrix is calculated as if the process were stationary. It has been shown by Parzano [13] that Ω_p is positive definite if and only if the process is stationary. However, the Cholesky factorization exists if and only if Ω_p is positive definite. Thus the algorithm fails outside the stationarity region.

In practice, the Martin and Wilkinson algorithm for decomposing Ω_p will encounter the square root of a negative number or a division by zero if Ω_p is not positive definite. The computer program can be written to give an error message in this case; an error message will be produced if and only if an attempt is made to evaluate a likelihood outside the region of stationarity.

Unfortunately, this does not hold for a mixed process. First, note that the transformations (3.1) and (4.3) are nonsingular and, thus, the matrix Ω_n^z which enters the decomposition algorithm is positive definite if and only if the matrix Ω_n is positive definite.

Now for sufficiently large k the autocovariances follow the difference equation:

$$\gamma_k = \phi_1\gamma_{k-1} + \cdots + \phi_p\gamma_{k-p} .$$

Because at least one root of $\phi(B)$ lies inside the unit circle, it is clear that for sufficiently large n $|\gamma_n| > \gamma_0$ and the resulting matrix will not be positive definite.

However, for any fixed value of n, it is possible that the process is nonstationary but the implied matrix is nevertheless positive definite. The following example demonstrates this. Consider the ARMA(1, 1) process:

$$y_t = \phi y_{t-1} + a_t - \theta a_{t-1} .$$

The covariance function is given by:

$$\gamma_0 = (1 - 2\theta\phi + \theta^2)/(1 - \phi^2)$$
$$\gamma_1 = (\phi - \theta)(1 - \theta\phi)/(1 - \phi^2)$$
$$\gamma_k = \phi^{k-1}\gamma_1 .$$

Suppose now that $\{w_t\}$ is nonstationary with parameter $\phi > 1$. We will show that for fixed n it is possible to choose θ so that the matrix Ω_n is positive definite. If we write $\theta = \phi - \alpha$ for $\alpha > 0$ we have:

$$\gamma_0 > 0 \qquad |\alpha| < \sqrt{\alpha^2 - 1}$$
$$|\gamma_1| = \alpha(\phi^2 - 1 - \phi\alpha) \qquad 0 < \alpha < \frac{\phi^2 - 1}{\phi}$$
$$= -\alpha(\phi^2 - 1 - \phi\alpha) \qquad \alpha \geq \frac{\phi^2 - 1}{\phi} .$$

For $1 < \phi < (1 + \sqrt{5})/2$, these functions take the form shown in Figure 1. (If $\phi > (1 + \sqrt{5})/2$, $(\phi^2 - 1)/\phi$ will fall to the right of $\sqrt{\phi^2 - 1}$.)

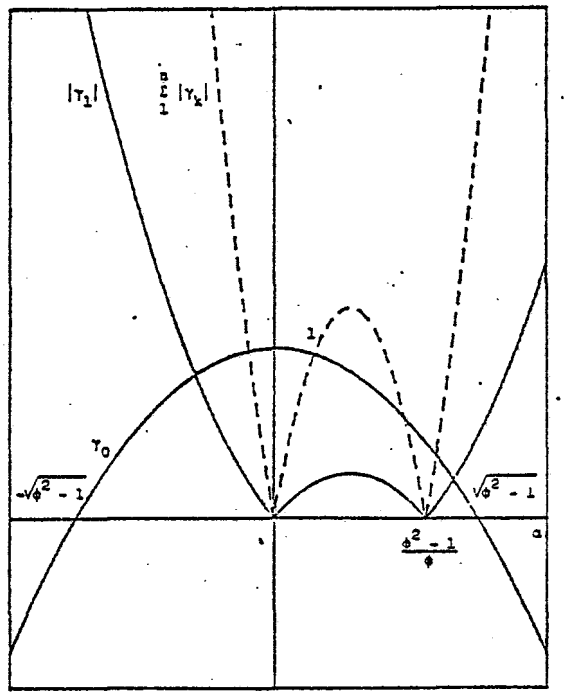


Fig. 1. Implied Covariances for Nonstationary ARMA(1, 1) Model

For $k \geq 1$, the $|\gamma_k|$'s form an increasing sequence with sum:

$$\sum_1^n |\gamma_k| = \frac{\phi^n - 1}{\phi - 1} |\gamma_1| \quad .$$

It is clear from Figure 1 that given any value of n we can choose α sufficiently small and positive that:

$$\gamma_0 > \sum_1^n |\gamma_k| > 0 \quad .$$

But, because the sequence $|\gamma_k|$ is increasing:

$$\sum_{j \neq i} |\Omega_{ij}^n| \leq \sum_1^n |\gamma_k| < \gamma_0 = \Omega_{ii}^n \quad i = 1, \ldots, n \quad .$$

It is well known that such a matrix is positive definite.

6. Invertibility and Maximum Likelihood

We show in this section that the likelihood is invariant under a change of representation in the moving average operator. This leads to a set of equivalent maxima at a number of points on the likelihood surface, each such point being the inverse in a sense of a unique maximum inside the invertible region. The sum of squares function, however, may not have minima outside the invertible region. If such minima do exist, they are not inverse to the least-squares point inside the invertible region.

Suppose that the model (1.1) is invertible, i.e., the roots of $\theta(B)$ lie outside the unit circle. We have:

$$\theta(B) = (1 - H_1 B)(1 - H_2 B), \ldots, (1 - H_q B) \qquad (6.1)$$

where $|H_k| < 1$, $k = 1, \ldots, q$. It is well known (see, for example, Box and Jenkins [2, p. 196]) that there are up to $2^q - 1$ corresponding non-invertible representations:

$$\phi(B) w_t = \theta^*(B) a_t \qquad (6.2)$$

obtained by inverting one or more of the roots of $\theta(B)$. It is understood that if any of the roots are complex they must be inverted in complex conjugate pairs. Suppose such a representation is obtained by inverting the first $m \leq q$ roots, i.e.,

$$\theta^*(B) = (1 - H_1^{-1} B) \cdot \ldots \cdot (1 - H_m^{-1} B) \cdot (1 - H_{m+1} B) \cdot \ldots \cdot (1 - H_q B) \quad . \qquad (6.3)$$

The variance of a_t^* in (6.2) is related to the variance of a_t in (1.1) by $\sigma_{a^*}^2 = k^2 \sigma_a^2$ where $k^2 = H_1^2 \cdot H_2^2 \cdot \ldots \cdot H_m^2 < 1$. The likelihood (1.2) can be restated as:

$$p(\underline{w}_n \underline{\phi}, \underline{\theta}, \sigma_{a^*}^2) = (2\pi\sigma_{a^*}^2)^{-n/2} |\Omega_n^*|^{-1/2} \cdot \exp\{-\underline{w}_n' \Omega_n^{*-1} \underline{w}_n / 2\sigma_{a^*}^2\} \qquad (6.4)$$

where $\sigma_{a^*}^2 \Omega_n^* = \sigma^2 \Omega_n$, i.e., $\Omega_n^* = \Omega_n / k^2$. The likelihood is invariant under the change of representation; this can be checked by direct substitution.

An immediate consequence of the invariance is that if there is a local maximum at $(\theta_1, \ldots, \theta_q)$ inside the invertible region, there are local maxima at each of the inverse points $(\theta_1^*, \ldots, \theta_q^*)$ defined by (6.3). There are up to $2^q - 1$ such inverse points, the number being smaller if equal or complex roots are present.

We can show this more explicitly by considering our algorithm for calculating the likelihood. Corresponding to (3.1) we can write:

$$\underline{w}_n' \Omega_n^{*-1} \underline{w}_n = \sum_1^n e_t^{*2}$$

where $e_t^* = k e_t$. The maximum likelihood estimators for σ_a^2 and $\sigma_{a^*}^2$ are

$$\hat{\sigma}_a^2 = \frac{1}{n} \sum_1^n e_t^2$$

$$\hat{\sigma}_{a^*}^2 = \frac{1}{n} \sum_1^n e_t^{*2} \quad .$$

Substituting into (1.2) and (6.4) respectively we obtain the log likelihoods maximized over σ_a^2 and $\sigma_{a^*}^2$.

$$L_{max} = c - \frac{n}{2} \ln S - \frac{1}{2} \ln |\Omega_n|$$

$$L_{max}^* = c - \frac{n}{2} \ln S^* - \frac{1}{2} \ln |\Omega_n^*|$$

where

$$S = \sum_1^n e_t^2, \quad S^* = \sum_1^n e_t^{*2} = k^2 S^2$$

$$c = -\frac{n}{2}[1 + \ln(2\pi/n)] \quad .$$

Because $\Omega_n^* = k^{-2} \Omega_n$ we have $|\Omega_n^*| = k^{-2n} |\Omega|$ and from (6.5) and (6.6) $L_{max} = L_{max}^*$. In practice, it is convenient to define:

$$\tilde{e}_t = |\Omega_n|^{1/2n} e_t = |\Omega_n^*|^{1/2n} e_t^*$$

to obtain

$$\tilde{L}_{max} = c - \frac{n}{2} \ln \sum_1^n \tilde{e}_t^2 \quad .$$

This is a suitable form for minimization by a non-linear least-squares algorithm.

To illustrate the invariance of the likelihood, we show the likelihood surface for an MA(1) process in Figure 2 and an MA(2) process in Figure 3.

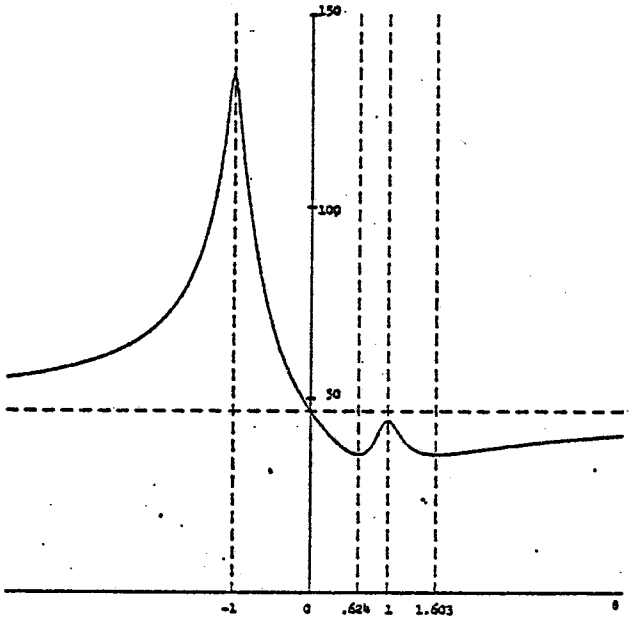


Fig. 2. Adjusted sum of squares $S = \sum_{1}^{n} \tilde{e}_t^2$ for MA(1) process

$w_t = a_t - .7a_{t-1} \quad \sigma_a^2 = 1 \quad n = 30$

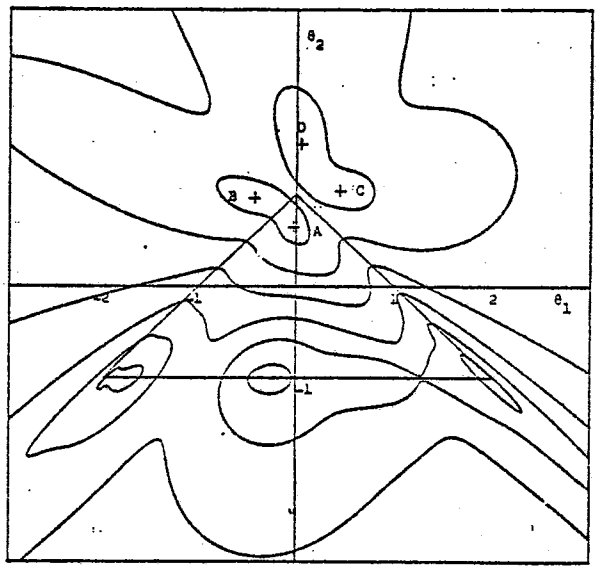


Fig. 3. Adjusted sum of squares $S = \sum_{1}^{n} \tilde{e}_t^2$ for MA(2) process

$w_t = a_t - .1a_{t-1} - .56a_{t-2} \quad \sigma_a^2 = 1 \quad n = 30$

Point A is the invertible minimum; B, C and D are noninvertible minima.

We turn now to the sum of squares function. If we write:

$$\theta^*(B) = (1 - G_1 B), \ldots, (1 - G_q B)$$

where

$$G_j = 1/H_j \qquad j = 1, \ldots, m$$
$$= H_j \qquad j = m + 1, \ldots, q$$

we have

$$S^*(G_1, \ldots, G_q)$$
$$= G_1^{-2}, \ldots, G_m^{-2} S(G_1^{-1}, \ldots, G_m^{-1}, G_{m+1}, \ldots, G_p) \ .$$

For $1 \leq j \leq m$:

$$\frac{\partial S^*}{\partial G_j} = - \prod_{\substack{i=1 \\ i \neq j}}^{m} G_i^{-2} \left[2G_j^{-3} S + G_j^{-4} \frac{\partial S}{\partial H_j} \right] \ .$$

If there is a minimum at $H_1, \ldots, H_q$ inside the invertibility region, it is clear that $\partial S^*/\partial G_j \neq 0$ and there is no corresponding minimum at an inverse point outside the invertibility region. Moreover, there can be a minimum outside the invertibility region only if:

$$H_j \frac{\partial S}{\partial H_j} + 2S = 0$$

at a point inside the invertibility region. Such a point may not exist. This is illustrated by Figure 4, which shows the sum of squares function for an MA(1) process; there is no minimum outside the invertibility region.

Aside from the above considerations, the numerical properties of existing methods make them unsatisfactory for a noninvertible representation. It can easily be shown that the backforecasting and conditional methods for approximating expected residuals diverge outside the invertible region. Whilst the Newbold/Dent approach is theoretically valid for a noninvertible representation, the nature of the matrix Z in (2.2) is such that it will become numerically unstable for long series.

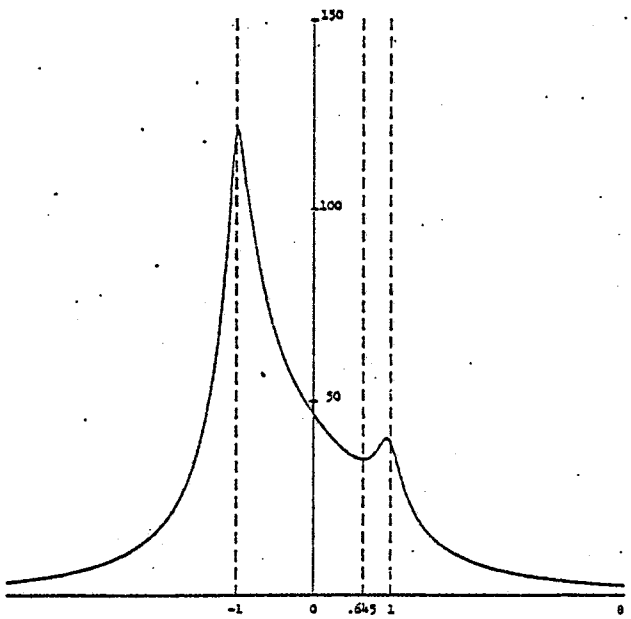


Fig. 4. Sum of squares function for MA(1) process

$w_t = a_t - .7a_{t-1} \quad \sigma_a^2 = 1 \quad n = 30$

7. Computational Notes

Because both the matrix L_n and the vector e_n in section 3 are calculated recursively, it is possible to combine the two phases of the calculation. A suitable modification of the Martin and Wilkinson algorithm has been developed by the author. For the low-order series usually encountered in practice, it has not been considered worthwhile to make use of all the structure in the matrix L_n for a seasonal moving average process. However, the zeroes in L_n have proved to be a major factor in improving efficiency.

To compare relative efficiencies, computer programs were also written for each of the three methods discussed earlier: conditional likelihood, the backforecasting approximation and the Newbold/Dent approach. For the backforecasting method, a single series of 100 backforecasts was calculated without further iteration. The Newbold/Dent method was programmed using the Golub and Reinsch [5] algorithm for singular value decomposition, and the Martin, Peters and Wilkinson [8] algorithm for triangular resolution of the matrix Σ.

Time comparisons for a number of model configurations with series length 100 were carried out on the DEC 20 computer at the University of Chicago Graduate School of Business. The results are shown in Table 1.

The relative advantage for the algorithm developed in this paper may be subject to minor change through refinements in coding. Also, the Golub-Reinsch algorithm involves iteration, and may therefore be dependent to some extent on parameter values. Nevertheless, the new method appears to have a clear advantage in efficiency over the other exact methods, especially for seasonal models.

TABLE 1

COMPARISON OF COMPUTING TIME

Approximate CPU Times in Milliseconds*

	Method			
Model	Conditional	Back-forecasting	Newbold/Dent	New Algorithm
ARMA(1, 0)	7	20	9	8
ARMA(0, 1)	9	21	53	28
ARMA(2, 0)	9	24	12	11
ARMA(0, 2)	9	27	174	43
ARMA(1, 1)	9	30	111	34
ARMA(1, 0).(1, 0)$_{12}$	14	39	1h1	111
ARMA(0, 1).(1, 0)$_{12}$	10	32	7,130	152
ARMA(1, 0).(0, 1)$_{12}$	12	22	1,550	174
ARMA(0, 1).(0, 1)$_{12}$	15	41	1,508	130

*Times for FORTRAN programs on DEC 20 computer at University of Chicago, Graduate School of Business. Series length n = 100.

REFERENCES

[1] Box, G. E. P.; Hillmer, S. C.; and Tiao, G.C. "Analysis and Modelling of Seasonal Time Series." Paper presented to the National Bureau of Economic Research/Bureau of the Census Conference on Seasonal Analysis of Economic Time Series, Washington, D.C., September 1976.

[2] Box, G. E. P., and Jenkins, G. M. Time Series Analysis, Forecasting and Control. Revised edition, San Francisco: Holden-Day, 1976.

[3] Dent, W. "Computation of the Exact Likelihood Function of an ARIMA Process." Journal of Statistical Computation and Simulation 5 (1977): 193-206.

[4] Dent, W., and Min, A-S. "A Monte Carlo Study of Autoregressive Integrated Moving Average Processes." Working Paper Series No. 76-12, Bureau of Business and Economic Research, University of Iowa, May 1976. (To appear in Journal of Econometrics.)

[5] Golub, G. H., and Reinsch, C. "Singular Value Decomposition and Least Squares Solutions." Numerische Mathematik 14 (1970): 403-20.

[6] Kang, K. M. "A Comparison of Estimators for Moving Average Processes." Australian Bureau of Statistics, 1973.

[7] McLeod, A. I. "Improved Box-Jenkins Estimators." Paper accepted February 1977 for publication in Biometrika.

[8] Martin, R. S.; Peters, G.; and Wilkinson, J. H. "Symmetric Decomposition of a Positive Definite Matrix." Numerische Mathematik 7 (1965): 362-83.

[9] Martin, R. S., and Wilkinson, J. H. "Symmetric Decomposition of Positive Definite Band Matrices." Numerische Mathematik 7 (1965): 355-61.

[10] Min, A-S. "A Study of Likelihood Functions of ARIMA Processes." Ph.D. thesis, University of Iowa, 1975.

[11] Nelson, C. R. "The First Order Moving Average Process: Identification, Estimation, and Prediction." Journal of Econometrics 2 (1974): 121-41.

[12] Newbold, P. "The Exact Likelihood Function for a Mixed Autoregressive-Moving Average Process." Biometrika 61 (1974): 423-26.

[13] Pagano, M. "When Is an Autoregressive Scheme Stationary?" Communications in Statistics 1 (1973): 533-44.

[14] Pierce, D. A. "Least Squares Estimation in the Regression Model with Autoregressive-Moving Average Errors." Biometrika 58 (1971): 299-312.

[15] Prothero, D. L. "Full Maximum Likelihood Estimation of Mixed ARMA Processes." Unpublished paper, London School of Economics, 1975.

SOFTWARE FOR RANK DEGENERACY

Gene Golub
Stanford University

Virginia Klema
MIT Center for Computational Research in
Economics and Management Science

ABSTRACT

We describe mathematical software that can be used to select a set of independent columns of a matrix when numerical rank can be determined, and we associate the uncertainty of the data with rank selection. The software is written in Fortran and is usable on a variety of computing machines.

This research was supported by the National Science Foundation under Grant MCS76-11989.

The software that we describe was designed initially to deal with problems of collinearity in data matrices associated with econometric modelling. Subsequently the software was used, in part, in the iteratively reweighted least squares problems in robust statistics and in the linear and nonlinear regression problems of data analysis. The need to determine numerical rank is present wherever row or column deletion is performed on data or design matrices, that is to say, whenever observations are deleted or when variable selections are made. Numerical rank is inherent in the problems of ridge regression and is implicit in the study of projection matrices, frequently referred to as "hat" matrices. Determination of numerical rank is also related to condition numbers associated with the solution of linear systems of equations, overdetermined or underdetermined systems of equations, and the iterations of linear systems that tend toward the solution of nonlinear least squares problems.

Rank determination in the presence of inexact arithmetic and uncertain data is far from trivial. Uncertainty in the data in the matrix, X, may lead to uncertainty in rank. If the data in, say, the third decimal place is known to be uncertain, that digit and all that follow are arbitrary . The matrix

$$\begin{bmatrix} 1.02 & 1.09 \\ 1.05 & 1.01 \end{bmatrix}$$

if uncertain in the third figure could lead to

$$\begin{bmatrix} 1.00 & 1.00 \\ 1.00 & 1.00 \end{bmatrix}.$$

Furthermore, the computational representation of X may be defective in rank when the exact representation of X has full rank, and conversely. One is therefore reduced to determining whether X is <u>near</u> a matrix that is defective in rank. That we must have precise and quantitative information on the "nearness" of one matrix to another

obviously influences our choice of matrix factorization methods.

Throughout this discussion we are assuming a linear model of the form $y = X\beta + e$ where y is an n-vector, and X is an n by p matrix. Classicially, if the data matrix X and the vector y are exact (that is to say, there is no uncertainty in the data X and y and both X and y can be represented exactly in the machine), if the precision of the arithmetic of the machine is such that X^TX (where X^T is the transpose of X) can be formed and stored exactly, and if X^TX is of full rank, the solution $\hat{\beta}$ could be obtained from $(X^TX)^{-1}X^Ty$. Cholesky factorization should then be used followed by the solution of the upper triangular systems of equations. However, these three conditions are frequently unattainable in practice, and we turn to other methods of matrix factorization, i.e., the norm-preserving orthogonal factorizations.

In the sequel we use the term "numerical rank" as described in [6], and if the n by p matrix X has numerical rank $k < p$, we seek k independent columns of X and identify $(p - k)$ dependent columns that are linear combinations of the k independent columns. Since the process we describe identifies relations among equations, a user may wish to specify initially that particular columns of X are essential for his model, and our software permits such an initial specification.

We begin the detection of rank degeneracy of the X matrix by forming the orthogonal factorization of X by Householder transformations as described in [1] and [9]. Any n by p matrix X can be factored in the form

$$X = QR$$

where the columns of Q are orthonormal, $Q^TQ = I$, and R is p by p and upper triangular. This factorization is unique to within signs of the rows of R and the corresponding columns of Q. For the model problem $y = X\beta + e$, $Q^Ty = c = Q^TX\beta = R\beta$, and β is the solution of the triangular system $R\beta = c$. Furthermore, the inverse matrix $(X^TX)^{-1} = (R^TR)^{-1} = R^{-1}R^{-T}$. If the X matrix is exactly rank deficient, the above factorization cannot be completed, and a reliable subroutine implementing the Householder factorization will indicate the number of columns of the X matrix that have been processed before rank deficiency occurred. If the subroutine is resistent to the side effects of underflow, and if column pivoting has been done, the number of columns of X that have been processed before termination indicates the number of independent (in some sense) columns of X. However the above determined number of indicated columns does not necessarily determine numerical rank in the sense that the number of columns obtained are <u>strongly</u> independent, which is to say that β may not be well defined by the transformation $X\beta = y$.

Unless X is exactly singular the completion of the QR factorization of X provides the upper triangular factor R whose column lengths and singular values are those of X. The singular value decomposition [5] of a matrix is one of the most elegant algorithms in numerical algebra for exposing quantitative information about the structure of a matrix. However the singular value decomposition of X where n is much greater than p is computationally expensive when compared to the Householder factorization. The singular values of X are the singular values of R, and if we need the singular value decomposition of the X matrix, we first do the Householder factorization and then compute the singular value decomposition of R.

Since we are concerned about rank degeneracy, and since frequently the data matrix X has some variables (columns) that are close in the numerical sense to being linear combinations of other columns we must detect this situation before we proceed with computations that could give erratic results that are less than meaningful. The numerical rank <u>should</u> be determined by the user or the originator of the problem, and this determination should be done with respect to the certainty of the data in the columns of X. Whenever inexact arithmetic

is used one is really solving a slightly perturbed problem involving the matrix (X + E) and the vectory y + δ.

Numerical rank determination is definitely scale-dependent, and scaling should reflect the user's best information about his problem. We propose multiplicative scaling of the columns of X and y such that the errors or perturbations induced in the representation of the elements of X and y are nearly the same. John Chambers has suggested such scaling strategy in [2]. Such scaling is also described in [6].

Before proceeding to compute the singular value decomposition we feel that it is important to get some additional information. Given the upper triangular matrix R, we know that its singular values are those of X, and the condition of R is that of X with respect to the solution of linear systems of equations or the solution of the least squares problem. Cline, Moler, Stewart, and Wilkinson [3] have provided a condition estimator involving the selection of a particular vector and the solution of two triangular systems of equations that gives a very good estimate of the largest and smallest singular values of R and hence of X. Since this condition estimator is computed in o (p^2) operations we always compute the estimates for R before deciding whether we should proceed with the singular value decomposition. If the condition estimate exceeds the certain digits of the data, or the reciprocal of the square root of the machine's precision, whichever is larger, we continue with the singular value decomposition.

The singular value decomposition of a matrix is a factorization of the matrix that exposes rank determination of X from the diagonal matrix Σ whose column lengths are those of X and R, the upper triangular factor of X. From the problem y = Xβ the singular value decomposition gives

$$U^T y = \Sigma V^T \beta$$

and the matrix X is explicitly $U\Sigma V^T$, where the columns of U are the orthonormal eigenvectors of XX^T, and the columns of V are the orthonormal eigenvectors of X^TX. The diagonal elements of Σ, $\sigma_1 \geq \sigma_2 \geq \sigma_3 \cdots \geq \sigma_n \geq 0$, are the singular values of X.

We now have exposed the structure of the matrix so that the determination of rank can be made by inspecting the elements of the diagonal matrix Σ. The number of independent columns of X is the number of independent columns of Σ, that is, the number of non-zero singular values of X. Given the uncertain digits in data matrices and computation with inexact arithmetic we still have the problem of determining which of the singular values are small and which are large. In practice we often see a well-defined gap in the singular values, and when such a gap occurs the choice of small singular values is obvious. The information contained in [6] gives a firm theoretical foundation for such a choice and gives additional information on choosing a stable set of independent columns of X. If there is no well-defined gap in the singular values it is more awkward to judge effective rank, but an example at the end of this discussion shows that some information is still obtainable.

Given rank, k, of Σ and, hence, of X, $\sigma_1 \geq \sigma_2 \geq \sigma_3 \cdots \geq \sigma_k \geq \sigma_{k+1} = \sigma_{k+2} \cdots = \sigma_p = 0$, and the solution $\hat{\beta}$ obtained from the singular value decomposition is

$$\hat{\beta} = V\Sigma^+ U^T y$$

where Σ^+ is the pseudo-inverse of Σ and is the diagonal matrix

$$\Sigma^+ = \begin{bmatrix} \frac{1}{\sigma_1} & & & & & \\ & \frac{1}{\sigma_2} & & & & \\ & & \ddots & & & \\ & & & \frac{1}{\sigma_k} & & \\ & & & & 0 & \\ & & & & & \ddots \\ & & & & & & 0 \end{bmatrix}$$

and we can effectively partion the matrix X as $[X_1 : X_2]$ where the columns of X_1 are linear combinations of the

columns of X_2, i.e., the columns of X_1 represent (p - k) near dependencies, and the columns of X_2 are the k independent columns of X.

We take advantage of the fact that in associating the columns of V_1 $(V_{k+1}, V_{k+2}, \ldots, V_p) = V_{null}$ corresponding to $(\sigma_{k+1}, \sigma_{k+2}, \ldots, \sigma_p)$

$$X V_{null} = 0$$

and a permutation of the rows of V_{null} corresponds to a permutation of the columns of X. This technique is not new - it is referred to in [7] and is theoretically and computationally justified in [6]. It has been used also to find the bases of free Jordan algebras [8].

To compute the indices of the columns of X_1 we form the Q R factorization with column pivoting of the rows of V_{null}. Explicitly we factor V_{null} as

$$Q^T V_{null}^T = [V_1 : V_2] .$$

V_1 is now upper triangular, nonsingular, and well-conditioned. The row indices thus determined identify column indices of the columns of X that belong in X_1. Alternatively, one could use stabilized elementary transformations to obtain the above factorization.

The information in the above factorization can be explored further to see whether the certainty of the data has been violated by the choice of rank of X.

$$X_1 V_1 + X_2 V_2 = 0$$
$$X_1 = -X_2 V_2 V_1^{-I}.$$

Let $G = -V_2 V_1^{-I}$ and $\tilde{G}$ be the modification of G to ignore elements that are negligably small. Then

$$(X_1 + \Delta_1) - (X_2 + \Delta_2) \tilde{G} = 0$$

where X_1, X_2, and $\tilde{G}$ are given and Δ_1 and Δ_2 are unknown and represent the perturbations induced in X_1 and X_2 by rank selection. Now

$$\Delta_1 - \Delta_2 \tilde{G} = -(X_1 - X_2 \tilde{G})$$

or

$$(\Delta_1, \Delta_2)\begin{pmatrix} I \\ -\tilde{G} \end{pmatrix} = -(X_1 - X_2 \tilde{G})$$

so that

$$(\Delta_1, \Delta_2) = -(X_1 - X_2 \tilde{G})\begin{pmatrix} I \\ -\tilde{G} \end{pmatrix}^+$$

and $\begin{pmatrix} I \\ -\tilde{G} \end{pmatrix}$ is of full rank.

In conclusion we show a small example that does not have a gap in singular values but, even so, one can get information on near-dependency among its columns.

The data matrix

$$\begin{bmatrix} 1.0101 & 1.0097 & .98 \\ 1.0098 & 1.0104 & .98 \\ .98 & .97 & 1.02 \\ .97 & .96 & .94 \\ .96 & .95 & .93 \end{bmatrix}$$

has singular values 3.7915, .007122, and .053328. We do not have a well-defined gap but there is some information to be gained from the vector, (-.732009, .678807, .058176), associated with the smallest singular value. The largest component in the vector would indicate that the first column should be deleted. The QR factorization with column pivoting of the matrix X has (-2.2052, .062485, and -0.01045) as the diagonal elements of R and has pivoted column 3 into the position of column 2. We believe the above example is a "worst case" type of problem from which we can still expose some information about numerical rank.

Acknowledgements

The authors thank Åke Björck for his suggestions for computing (Δ_1, Δ_2), and we thank Neil Kaden, David Coleman, and Steve Peters for programming assistance. We acknowledge PFORT, Bell Labs., in checking portability of the software and the IMSL converter to generate target code for non-IBM machines. The software for the singular value decomposition is from EISPACK II [4].

We gratefully acknowledge the support of the National Science Foundation for this research.

References

[1] Businger, P., and Golub, G.H., "Linear Least Squares Solutions by Householder Transformations," Numer. Math. 1 (1965), 269-276.

[2] Chambers, J.M., "Stabilizing Linear Regression Against Observational Error in Independent Variates," unpublished manuscript, Bell Labs., Murray Hill, New Jersey, (1972).

[3] Cline, A.K., Moler, C.B., Stewart, G.W., and Wilkinson, J.H., "An Estimate for the Condition Number of a Matrix," ANL TM-310, (1977).

[4] Garbow, B.S., Boyle, J.M., Dongarra, J.J., and Moler, C.B., Matrix Eigensystem Routines-EISPACK Guide Extension, Springer-Verlag, Lecture Notes in Computer Science, 51, (1977).

[5] Golub, G.H. and Kahan, W., "Calculating the Singular Values and Pseudo-inverse of a Matrix," J. SIAM Numer. Anal. SER B 2, 205-224, (1965).

[6] Golub, G.H., Klema, V., and Stewart, G.W., "Rank Degeneracy and Least Squares Problems," University of Maryland, TR-456, (1976), Stanford University, STAN-C5-76-559, (1976), National Bureau of Economic Research, Inc. Working Paper 165, (1977).

[7] Lawson, C.L. and Hanson, R.J., Solving Least Squares Problems, Prentice-Hall, (1974).

[8] Smith, Brian, (private communication, 1975).

[9] Stewart, G.W., Introduction to Matrix Computations, Academic Press, (1973).

TEST PROBLEMS AND TEST PROCEDURES FOR LEAST SQUARES ALGORITHMS

Roy H. Wampler
National Bureau of Standards
Washington, D. C. 20234

Abstract

Numerous test problems have been introduced in the past twenty years for the purpose of studying and comparing least squares algorithms and computer programs. This paper discusses and classifies some of the useful test problems which have appeared in the literature. A recent large-scale test procedure is briefly summarized. Several neat, mathematical examples are displayed. A new example is presented, and results from several computer programs in solving this problem are given.

1. INTRODUCTION

In the past twenty years, numerous test problems have been introduced for the purpose of testing and comparing linear least squares algorithms and computer programs. Such test problems have proved to be useful in comparing two or more algorithms with respect to numerical accuracy and computational efficiency. They have sometimes been used in comparative evaluations of least squares regression programs. They can be used to study the properties of a particular algorithm or to compare the performance of a particular program on two or more types of computers.

In this paper, many of the least squares test problems which have appeared in the literature are classified in several ways. Three problems of the neat, mathematical type are displayed. A new test problem is given, and the performance of several algorithms in solving this problem is discussed.

For recent accounts of the state of the art in solving least squares problems, the reader is referred to Linear Regression Analysis by Seber (1977), especially Chapter 11 on computational techniques, to Computational Methods for Data Analysis by Chambers (1977), especially Chapter 5 on linear models and the appendix which lists some available algorithms, and to Solving Least Squares Problems by Lawson and Hanson (1974).

Gregory and Karney (1969) gave a collection of numerical examples related to matrix inversion, systems of linear equations, eigenvalues and eigenvectors.

2. FORMULATION OF THE LEAST SQUARES PROBLEM

The simplest form of the linear least squares problem can be stated as follows: Given a matrix X ($n \times p$) and a vector of observations y ($n \times 1$), find a vector of coefficients $\hat{\beta}$ ($p \times 1$) which minimizes the sum of squares of the residual vector $\delta = y - X\beta$. When X is of rank p, the unique solution can be expressed as

$$\hat{\beta} = (X'X)^{-1}X'y.$$

Other quantities of interest are $\hat{y} = X\hat{\beta}$, the vector of predicted values, and $(X'X)^{-1}$, the unscaled covariance matrix of the coefficients.

A method frequently used in econometric work for obtaining the solution vector and related quantities requires the forming and inverting of a matrix of correlation coefficients. In this

method it is assumed that one column of X, say the first, has each element equal to one. Let

$$\bar{x}_j = \frac{\sum_i x_{ij}}{n}, \qquad \bar{y} = \frac{\sum_i y_i}{n},$$

$$z_{ij} = x_{ij} - \bar{x}_j, \qquad u_i = y_i - \bar{y},$$

$$s_j = (\sum_i z_{ij}^2)^{\frac{1}{2}}, \qquad s_y = (\sum u_i^2)^{\frac{1}{2}},$$

$$r_{jk} = \frac{\sum_i z_{ij} z_{ik}}{s_j s_k}, \qquad r_{jy} = \frac{\sum_i z_{ij} u_i}{s_j s_y}$$

$R_{xx} = (r_{jk})$ for $j=2,3,\ldots,p$; $k=2,3,\ldots,p$;

$$r_{xy} = \begin{bmatrix} r_{2y} \\ r_{3y} \\ \cdot \\ \cdot \\ \cdot \\ r_{py} \end{bmatrix}.$$

The $(p-1)\times(p-1)$ matrix R_{xx} is known as a correlation matrix.

If we let

$$a = R_{xx}^{-1} r_{xy} = \begin{bmatrix} a_2 \\ a_3 \\ \cdot \\ \cdot \\ \cdot \\ a_p \end{bmatrix}$$

and $\hat{\beta} = \begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_p \end{bmatrix}$,

the desired coefficients can be obtained from the formulas

$$b_j = \frac{a_j s_y}{s_j} \quad \text{for } j=2,3,\ldots,p,$$

and $b_1 = \bar{y} - b_2\bar{x}_2 - b_3\bar{x}_3 - \ldots - b_p\bar{x}_p$.

Formulas for expressing the diagonals of $(X'X)^{-1}$ in terms of R_{xx}^{-1}, $\bar{x}_j$ and s_j are given in Seber (1977) and Horiba (1971).

Maindonald (1977), in his article discussing least squares computations based on the correlation matrix, observes that this method is well adapted to updating, as in stepwise regression.

3. CLASSIFICATION OF TEST PROBLEMS

Least squares test problems can be classified in a variety of ways. Below are given several categories of test problems with examples of these types which have appeared in the literature. In most cases, either the data for the test problems were published or an algorithm was given for generating the data. In a few instances, however, the data were not published.

1. (a) Problems arising from real data. Examples: Freund (1963); Zellner and Thornber (1966); Longley (1967); Krane (1970, 1971); Chambers (1971); Mullet and Murray (1971); Chambers (1973); Gentleman (1975); Beaton, Rubin and Barone (1976); Longley (1976); Chambers (1977).

(b) Problems constructed from contrived data. Examples: Läuchli (1961); Bauer (1965); Golub and Reinsch (1970); Stewart (1973); Lawson and Hanson (1974); Kenkel (1976); etc.

2. (a) Polynomial regression problems. Examples: Cameron (1957); Ascher and Forsythe (1958); Macdonald (1964); Bright and Dawkins (1965); Jordan (1968); Muhonen (1968); Wampler (1969, 1970); Jennings and Osborne (1974); Fletcher (1975); Shampine (1975); Boehm, Mankhaus and Penn (1976); Longley (1976); Björck (1978).

(b) Multiple regression problems. Examples: Bauer (1965); Golub (1965); Zellner and Thornber (1966); Longley (1967); etc.

3. Problems having known condition number, or known to be ill-conditioned. Segments of the Hilbert matrix or of its inverse have been used by Golub (1965); Businger and Golub (1965); Golub and Wilkinson (1966); Björck and Golub (1967); Björck (1968); Peters and Wilkinson (1970); Abdelmalek (1971); Plemmons (1974); Daniel et al. (1976); Björck (1978). Björck and Golub (1973) used Vandermonde matrices.

4. Problems which are mathematically rank-deficient. Examples: Lawson and Hanson (1974), p. 277 and p. 311, using a Fortran function to generate data; Velleman, Seaman and Allen (1977).

5. Problems constructed from a set of orthogonal vectors. Example: Hastings (1972).

6. Problems constructed from the singular value decomposition. Examples: Velleman, Seaman and Allen (1977); Seaman (1977).

7. Problems having special features, such as:

(a) Solution vector is subject to linear constraints. Examples: Björck and Golub (1967); Björck (1968); Stoer (1971); Hastings (1972); Gerig and Gallant (1975).

(b) Observations have unequal variances. Examples: Shampine (1975); Björck (1978); Wampler (submitted).

(c) Data matrix X is sparse. Example: Gentleman (1975).

(d) Solutions are known for several regression methods, as least squares, least absolute residuals, and other robust methods. Example: Holland (1976).

4. A RECENT LARGE-SCALE STUDY

Velleman, Seaman and Allen (1977) recently reported the results of a large-scale evaluation of least squares regression routines available in widely distributed statistical packages. Velleman and co-workers (1975, 1977) devised a test procedure which considered five factors that can lead to computational difficulties:

(1) The columns of X can become highly collinear.

(2) A column of X can have a coefficient of variation which approaches zero.

(3) R^2, the coefficient of determination, can approach zero, indicating failure of the statistical model.

(4) The absolute magnitude of the numbers in X can become large, defeating absolute checks for rank-deficiency.

(5) The ratio of the magnitude of y to that of X can vary widely.

The procedure which was developed by Velleman et al. for probing these five factors generated sets of data through use of the singular value decomposition of the matrix X. Some of the generated data sets were mathematically rank-deficient. The seven statistical package regression routines which were evaluated displayed considerable variation in the accuracy obtained as well as in the protection given to users against computational disasters.

5. NEAT, MATHEMATICAL TEST PROBLEMS

Lauchli (1961) introduced the following test problem:

$$X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ \varepsilon & 0 & 0 & 0 & 0 \\ 0 & \varepsilon & 0 & 0 & 0 \\ 0 & 0 & \varepsilon & 0 & 0 \\ 0 & 0 & 0 & \varepsilon & 0 \\ 0 & 0 & 0 & 0 & \varepsilon \end{bmatrix}, \quad y = \begin{bmatrix} \varepsilon \\ 0 \\ -5 \\ 5 \\ -5 \\ 0 \end{bmatrix}.$$

In the matrix X'X all diagonal terms equal $1 + \varepsilon^2$ and all off-diagonal terms equal 1. For $\varepsilon \neq 0$ the rank of X'X is five since the eigenvalues are $5 + \varepsilon^2, \varepsilon^2, \varepsilon^2, \varepsilon^2, \varepsilon^2$. Golub (1965) noted that if η is the largest number on the computer such that $1.0 + \eta = 1$ in floating point arithmetic, then whenever $\varepsilon < \sqrt{\eta}/2$ the rank of the computed matrix X'X will be one. No matter how accurate the linear equation solver, it is impossible to solve the normal equations $X'X\beta = X'y$.

This problem (sometimes with variations) has been widely used by later authors — e.g., Björck (1967), Dahlquist and Björck (1974).

Stewart (1973), p. 228, gave the matrix

$$X = \begin{bmatrix} 1 & 1 + \varepsilon \\ 1 & 1 - \varepsilon \\ 1 & 1 \end{bmatrix}$$

as an exercise in computing an upper bound on the condition number of X as a function of ε.

Lawson and Hanson (1974), p. 127, used the matrix

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 - \varepsilon \end{bmatrix}$$

in comparing the accuracy obtainable from Cholesky decomposition with that obtainable from Householder transformations. They observed that when one works with a fixed precision, a larger class of problems can be solved using Householder transformations than can be solved by forming the normal equations and using the Cholesky decomposition.

The three examples displayed above are quite useful pedagogically, but they are not suitable for comparing the correlation matrix method with other methods for obtaining least squares solutions. (This is because Lauchli's example lacks a constant column vector and the other two examples lead to a correlation matrix having but one element, $R_{xx} = [1]$.)

6. A NEW TEST PROBLEM

It is sometimes claimed that when one obtains least squares solutions via the correlation matrix, in which the raw data are centered about their means and divided by their standard deviations, the numerical instability which can arise in solving the normal equations is eliminated. One may ask: How does the correlation matrix method compare with orthogonalization methods (such as Householder transformations or modified Gram-Schmidt) which work directly with the data matrix X?

Consider the following example. Let

$$X = \begin{bmatrix} 1 & 1+\varepsilon & 1-\varepsilon \\ 1 & 2+\varepsilon & 2-\varepsilon \\ 1 & 2-\varepsilon & 2+\varepsilon \\ 1 & 1-\varepsilon & 1+\varepsilon \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

where $\varepsilon > 0$ and the y_i are any real numbers. The matrices R_{xx}, R_{xx}^{-1} and $(X'X)^{-1}$ for this example are:

$$R_{xx} = \begin{bmatrix} 1 & \frac{1-4\varepsilon^2}{1+4\varepsilon^2} \\ \frac{1-4\varepsilon^2}{1+4\varepsilon^2} & 1 \end{bmatrix},$$

$$R_{xx}^{-1} = \frac{1+4\varepsilon^2}{16\varepsilon^2} \begin{bmatrix} 1+4\varepsilon^2 & -1+4\varepsilon^2 \\ -1+4\varepsilon^2 & 1+4\varepsilon^2 \end{bmatrix}$$

and

$$(X'X)^{-1} = \begin{bmatrix} \frac{5}{2} & -\frac{3}{4} & -\frac{3}{4} \\ -\frac{3}{4} & \frac{1}{4}+\frac{1}{16\varepsilon^2} & \frac{1}{4}+\frac{1}{16\varepsilon^2} \\ -\frac{3}{4} & \frac{1}{4}-\frac{1}{16\varepsilon^2} & \frac{1}{4}+\frac{1}{16\varepsilon^2} \end{bmatrix}.$$

Taking $\varepsilon = 10^{-k}$, $k=0,1,2,\ldots$ and

$$y = \begin{bmatrix} 0 \\ 4 \\ 2 \\ 2 \end{bmatrix}$$

one obtains

$$\hat{\beta} = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} \quad \text{and} \quad \delta = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$$

for all values of ε. The diagonals d_j of $(X'X)^{-1}$ which one needs for obtaining standard deviations of the estimated coefficients, are $d_1 = 5/2$, $d_2 = d_3 = \frac{10^{2k}+4}{16}$.

I have carried out numerical experiments on this problem using computer programs based on the following four methods:

(1) Inversion of the matrix of correlation coefficients using the subroutines SYMINV and CHOL of Healy (1968) for inverting a symmetric semi-definite matrix via Cholesky decomposition.

(2) Householder transformation algorithm using my Fortran version of the Björck-Golub (1967) algorithm, with iterative refinement of the solution omitted.

(3) Modified Gram-Schmidt algorithm using L2A subroutine of Wampler (submitted), with iterative refinement of the solution omitted.

(4) Like (3), but with iterative refinement included.

The computing was done on a Univac 1108. All computations were done in single precision arithmetic (approximately 8 decimal digits) except that program (4) accumulated inner products in double precision.

Selected results, for $\varepsilon = 10^{-2}$, 10^{-4} and 10^{-5}, are given in Table 1 for computed coefficients and in Table 2 for computed diagonal terms of $(X'X)^{-1}$.

We note that as ε becomes smaller, the accuracy of certain computed results deteriorates. All four programs found the input matrix to be of full rank for $\varepsilon \geq 10^{-7}$ but reported rank deficiency for $\varepsilon = 10^{-8}$.

The accuracy of programs (1), (2) and (3) is about the same for coefficients b_2 and b_3. In the case of b_1, the "intercept" term, program (1) which used the correlation matrix method obtained poor accuracy compared with programs (2) and (3). Program (4), which uses iterative refinement, gives information on the behavior of the refinement procedure. The fact that the initial solutions for $\varepsilon = 10^{-4}$ and 10^{-5} were estimated to have less than one correct digit is a clear indication of the ill-conditioning of the problem.

TABLE 1. COMPUTED SOLUTION VECTORS FROM FOUR PROGRAMS

		$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$
PROGRAM 1. Inversion of Correlation Matrix	b_1	-0.99998595	-0.65165032	-1.1819804
	b_2	0.99999532	1.0606601	0.70710677
	b_3	0.99999532	0.70710677	1.4142135
PROGRAM 2. Householder Transformations	b_1	-0.99999997	-0.99999993	-0.99999916
	b_2	1.0000200	0.74929179	-17.810433
	b_3	0.99998000	1.2507082	19.810433
PROGRAM 3. Modified Gram-Schmidt	b_1	-1.0000000	-1.0000000	-0.99999999
	b_2	0.99999954	0.90688419	28.936355
	b_3	1.0000005	1.0931158	-26.936357
PROGRAM 4. Modified Gram-Schmidt with Iterative Refinement	b_1	-1.0000000	-1.0000000	-0.99999998
	b_2	0.99999070	0.81372793	19.617495
	b_3	1.0000093	1.1862721	-17.617495
Number of iterations		2	3	4
Est. correct digits in initial solution		5.1	0.8	0.3

TABLE 2. COMPUTED DIAGONALS OF $(X'X)^{-1}$ FROM THREE PROGRAMS

		$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$
PROGRAM 1. Inversion of Correlation Matrix	d_1	2.5000000	2.3750000	2.0000000
	d_2	625.25195	7456540.1	13421772.
	d_3	625.25195	7456540.1	13421772.
PROGRAM 2. Householder Transformations	d_1	2.5000000	2.5000001	2.5000013
	d_2	625.25026	6249792.2	623761330.
	d_3	625.25024	6249792.3	623761340.
PROGRAM 3. Modified Gram-Schmidt	d_1	2.5000001	2.4999999	2.5000000
	d_2	625.24962	6250486.1	624926620.
	d_3	625.24959	6250486.6	624926670.
Exact Solution	d_1	2.5000000	2.5000000	2.5000000
	d_2	625.25000	6250000.25	625000000.25
	d_3	625.25000	6250000.25	625000000.25

In the case of the diagonal terms of $(X'X)^{-1}$, i.e., the quantities needed for obtaining standard deviations of the computed coefficients, the accuracy of program (1) was noticeably poorer than that of programs (2) and (3) for the smaller values of ε.

7. REFERENCES

Abdelmalek, Nabih N., Round off error analysis for Gram-Schmidt method and solution of linear least squares problems, BIT, Vol. 11, pp. 345-367 (1971).

Ascher, Marcia, and George E. Forsythe, SWAC experiments on the use of orthogonal polynomials for data fitting, Journal of the Association for Computing Machinery, Vol. 5, pp. 9-21 (1958).

Bauer, F.L., Elimination with weighted row combinations for solving linear equations and least squares problems, Numerische Mathematik, Vol. 7, pp. 338-352 (1965).

Beaton, Albert E., Donald B. Rubin, and John L. Barone, The acceptability of regression solutions: Another look at computational accuracy, Journal of the American Statistical Association, Vol. 71, pp. 158-168 (1976).

Björck, Ake, Solving linear least squares problems by Gram-Schmidt orthogonalization, BIT, Vol. 7, pp. 1-21 (1967).

Björck, Ake, Iterative refinement of linear least squares solutions, II, BIT, Vol. 8, pp. 8-30 (1968).

Björck, Ake, Comment on the iterative refinement of least squares solutions, to appear in Journal of the American Statistical Association (1978).

Björck, Ake and Gene Golub, ALGOL Programming, Contribution No. 22: Iterative refinement of linear least square solutions by Householder transformation, BIT, Vol. 7, pp. 322-337 (1967).

Björck, Ake, and Gene H. Golub, Numerical methods for computing angles between linear subspaces, Mathematics of Computation, Vol. 27, pp. 579-594 (1973).

Boehm, William T., D.J. Menkhaus and J.B. Penn, Accuracy of least squares computer programs: Another reminder, American Journal of Agricultural Economics, Vol. 58, No. 4, Part I, pp. 757-760 (1976).

Bright, J.W., and G.S. Dawkins, Some aspects of curve fitting using orthogonal polynomials, Industrial and Engineering Chemistry Fundamentals, Vol. 4, pp. 93-97 (1965).

Businger, Peter and Gene H. Golub, Linear least squares by Householder transformations, Numerische Mathematik, Vol. 7, pp. 269-276 (1965).

Cameron, J.M., Some examples of the use of high speed computers in statistics, Proceedings of the First Conference on the Design of Experiments in Army Research, Development and Testing, Office of Ordnance Research, Report No. 57-1, pp. 129-135 (1957).

Chambers, John M., Regression updating, Journal of the American Statistical Association, Vol. 66, pp. 744-748 (1971).

Chambers, John M., Linear regression computations: Some numerical and statistical aspects, Bulletin of the International Statistical Institute, Proceedings of the 39th Session, Vienna, Vol. 45, Book 4, pp. 245-254 (1973).

Chambers, John M., Computational Methods for Data Analysis, Wiley, New York (1977).

Dahlquist, Germund, and Ake Björck, Numerical Methods (translated by Ned Anderson), Prentice-Hall, Englewood Cliffs, N.J. (1974).

Daniel, J.W., W.B. Gragg, L. Kaufman, and G.W. Stewart, Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization, Mathematics of Computation, Vol. 30, pp. 772-795 (1976).

Fletcher, R.H., On the iterative refinement of least squares solutions, Journal of the American Statistical Association, Vol. 70, pp. 109-112 (1975).

Freund, R.J., A warning of roundoff errors in regression, The American Statistician, Vol. 17, No. 5, pp. 13-15 (1963).

Gentlemen, W. Morven, Row elimination for solving sparse linear systems and least squares problems, in Numerical Analysis, Proceedings of the Dundee Conference on Numerical Analysis, edited by G.A. Watson, Springer-Verlag, Berlin/Heidelberg/New York, pp. 122-133 (1975).

Gerig, Thomas M., and A Ronald Gallant, Computing methods for linear models subject to linear parametric constraints, Journal of Statistical Computation and Simulation, Vol. 3, pp. 283-296 (1975).

Golub, G., Numerical methods for solving linear least squares problems, Numerische Mathematik, Vol. 7, pp. 206-216 (1965).

Golub G.H., and C. Reinsch, Singular value decomposition and least squares solutions, Numerische Mathematik, Vol. 14, pp. 403-420 (1970).

Golub, G.H., and J.H. Wilkinson, Note on the iterative refinement of least squares solution, Numerische Mathematik, Vol. 9, pp. 139-148 (1966).

Gregory, Robert T. and David L. Karney, A Collection of Matrices for Testing Computational Algorithms, Wiley-Interscience, New York (1969).

Hastings, W. Keith, Test data for statistical algorithms: Least squares and ANOVA, Journal of the American Statistical Association, Vol. 67, pp. 874-879 (1972).

Healy, M.J.R., Algorithm AS 6: Triangular decomposition of a symmetric matrix; Algorithm AS 7: Inversion of a positive semi-definite symmetric matrix. Applied Statistics, Vol. 18, pp. 195-199 (1968). (Corrigenda, Applied Statistics, Vol. 18, p. 118 (1969)).

Hocking, Ralph T., Analysis of factors using multiple regression, Journal of Systems Management, Vol. 21, pp. 29-34 (1970).

Holland, Paul W., Robust test problems for robust regression programs, Proceedings of the Ninth Interface Symposium on Computer Science and Statistics, Prindle, Weber and Schmidt, Inc., Boston, Mass., pp. 99-110 (1976).

Horiba, Y., A note on an efficient computation of the standard error of the constant term in multiple regression analysis (Letter to the Editor), The American Statistician, Vol. 25, No. 4, pp. 51-52 (1971).

Jennings, L.S., and M.R. Osborne, A direct error analysis for least squares, Numerische Mathematik, Vol. 22, pp. 325-332 (1974).

Jordan, T.L., Experiments on error growth associated with some linear least-squares procedures, Mathematics of Computation, Vol. 22, pp. 579-588 (1968).

Kenkel, J.L., Some methods for examining rounding error in a least squares regression computer program, Journal of Economics and Business, Vol. 28, pp. 104-110 (1976).

Krane, Scott A., Letter to the Editor, The American Statistician, Vol. 24, No. 4, p. 46 (1970). Letter to the Editor, The American Statistician, Vol. 25, No. 3, p. 52 (1971). (An article by Hocking (1970) is discussed.)

Läuchli, Peter, Jordan-Elimination und Ausgleichung nach kleinsten Quadraten, Numerische Mathematik, Vol. 3, pp. 226-240 (1961).

Lawson, Charles L., and Richard J. Hanson, Solving Least Squares Problems, Prentice-Hall, Englewood Cliffs, N.J. (1974).

Longley, James W., An appraisal of least squares programs for the electronic computer from the point of view of the user, Journal of the American Statistical Association, Vol. 62, pp. 819-841 (1967).

Longley, James W., A subroutine for singular value decomposition by normalized modified Gram-Schmidt algorithm for the solution of linear least squares equations, Proceedings of the Statistical Computing Section, American Statistical Association, Washington, D.C., pp. 207-212 (1976).

Maindonald, J.H., Least squares computations based on the Cholesky decomposition of the correlation matrix, Journal of Statistical Computation and Simulation, Vol. 5, pp. 247-258 (1977).

Macdonald, J. Ross, Accelerated convergence, divergence, iteration, extrapolation, and curve fitting, Journal of Applied Physics, Vol. 35, pp. 3034-3041 (1964).

Muhonen, Daniel P., An experimental comparison of several approaches to the linear least squares problem, Report TR-01-1004, Bendix Field Engineering Corporation, prepared under Contract NAS5-10750 for National Aeronautics and Space Administration, Goddard Space Flight Center, Greenbelt, Maryland, Nov. 1968.

Mullet, Gary M. and Tracy W. Murray, A new method for examining rounding error in least-squares regression computer programs, Journal of the American Statistical Association, Vol. 66, pp. 496-498 (1971).

Peters, G. and J.H. Wilkinson, The least squares problem and pseudo-inverses, The Computer Journal, Vol. 13, pp. 309-316 (1970).

Plemmons, Robert J., Linear least squares by elimination and MGS, Journal of the Association for Computing Machinery, Vol. 21, pp. 581-585 (1974).

Seaman, Jeff, Use of computer resources by statistical package regression programs, Proceedings of the Statistical Computing Section, American Statistical Association, Washington, D.C. (1977).

Seber, G.A.F., Linear Regression Analysis, Wiley, New York (1977).

Shampine, L.F., Discrete least squares polynomial fits, Communications of the ACM, Vol. 18, pp. 179-180 (1975).

Stewart, G.W., Introduction to Matrix Computations, Academic Press, New York (1973).

Stoer, Josef, On the numerical solution of constrained least-squares problems, SIAM Journal on Numerical Analysis, Vol. 8, pp. 382-411 (1971).

Velleman, Paul F., and Ivor Francis, Measuring statistical accuracy of regression programs, Proceedings of the Computer Science and Statistics Eighth Annual Symposium on the Interface, edited by James W. Frane, University of California, Los Angeles, pp. 122-127 (1975).

Velleman, Paul F., Jeff Seaman and I. Elaine Allen, Evaluating package regression routines, Proceedings of the Statistical Computing Section, American Statistical Association, Washington, D.C. (1977).

Wampler, Roy H., An evaluation of linear least squares computer programs, Journal of Research of the National Bureau of Standards, Vol. 73B, pp. 59-90 (1969).

Wampler, Roy H., A report on the accuracy of some widely used least squares computer programs, Journal of the American Statistical Association, Vol. 65, pp. 549-565 (1970).

Wampler, Roy H., Solutions to weighted least squares problems by modified Gram-Schmidt with iterative refinement (submitted for publication).

Zellner, A., and H. Thornber, Computational accuracy and estimation of simultaneous equation econometric models, Econometrica, Vol. 34, pp. 727-729 (1966).

WORKSHOP 4

GRAPHICS

Chair: Ronald K. Lohrding, Los Alamos Scientific Laboratory

Portable Graphical Software for Data Analysis

Richard A. Becker

Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

Three distinct problems arise in providing a library of portable graphical routines for data analysis. First, the language used must be widely available and standardized. Second, the routines must deal with a wide variety of graphics devices. Finally, operating system considerations will affect overall portability.

These problems may be dealt with by using a language such as FORTRAN, by providing a standard device interface, and by isolating operating system dependencies in small primitive modules. This approach has been implemented at Bell Laboratories in the GR-Z graphical system, which is currently running on a number of different computing systems.

1. Background

Graphical techniques are an integral part of the field of data analysis, and new forms of graphical displays are an important product of current statistical research. Much of this new work utilizes computer graphics. Portable graphical software (which is easily moved from one computing environment to another), provides an excellent means of communicating the details of the plotting technique.

Unlike most algorithms, graphical routines operate with specialized output devices that are not fully supported by the operating system. In addition, the rapid growth of the hardware end of the graphics field makes it important for graphical routines to be moved from one device to another without rewriting. Since the computing environment encompasses the output device, graphical routines have extra problems of portability.

Three basic approaches to providing graphical software use either: (1) an existing programming language which incorporates graphical operations, (2) a preprocessor to add graphics to a non-graphic language, or (3) a subroutine library in conjunction with an existing language. The first approach is limited; there are few languages with built-in graphical primitives, and they often provide only limited facilities [Luehrmann]. Provision of software useful for data analysis within the context of such a language still requires a subroutine library to provide high-level data-analytic operations.

The second approach can provide improved syntax, but must eventually rely on support routines. Therefore, it too is dependent on the subroutine library approach.

The third method, using an existing general-purpose language augmented by a subroutine library of graphical operations, provides the advantages of flexibility and general ease of implementation, and allows a choice of currently available and familiar languages. Since the other two approaches require support libraries, this method appears to be the most basic. Consequently, the remainder of this paper will assume a subroutine library approach to providing portable graphical software.

Run-time device support routines are required for any graphical system, and these routines often require interaction with the operating system. With a language, run-time routines are normally provided on each machine to support specific graphics devices. The interface with the operating system is provided by the compiler writer (often the manufacturer of the hardware.)*

With a subroutine library, it is often difficult to provide appropriate operating system interactions. One way this may be accomplished is by postulating a basic set of graphical subroutines to be implemented on each machine [SIGGRAPH]. Another approach assumes a small set of primitive system-related routines, and tries to provide machine independent routines for the rest of the system. This method will be explored further in the following sections.

2. Portable Graphical Subroutines

There are several dimensions to the problem of providing portable graphical subroutines. Perhaps most obvious (and best understood) is the problem of writing in a computer language that is readily available on different machines and is sufficiently standardized to give predictable results.

Another problem of portability involves moving of graphical software from one device to another. The problems of accomplishing this are less-well understood than for language portability.

* This sounds ideal, but note, however, the need to rely on the manufacturer to provide support for new devices, etc.

Finally, since subroutine libraries have no special ability to interface with the operating system, certain problems arise from interactions with the operating system. Portable subroutines must still go through machine-dependent processes such as library set-up and search. In addition, device control and character set problems may also manifest themselves.

The next three sections will discuss these three areas in more detail.

2.1. Language Aspects of Portability

The merits of choosing one programming language over another for the purpose of graphics could be debated endlessly. However, FORTRAN seems to be a reasonable choice due to its wide availability, its ability to have separately compiled routines, and a fairly wide body of experience regarding the portability of FORTRAN.

One important feature of FORTRAN is the existance of a well-defined portable subset of the language along with a software tool to ensure adherence to the subset. The portable subset of Standard FORTRAN [ANSI] is known as PFORT [Ryder]. The PFORT Verifier is a (portable) FORTRAN program which checks for adherence to this rather restrictive language. Not only can the PFORT Verifier be used to check the portability of the graphics system routines, but it can also check the portability of applications routines that use the graphics system.

All would be well on the language front if it were possible to stick strictly to PFORT, but, unfortunately, this is impossible if a useful graphical system is to result. The problem arises because of FORTRAN's limited treatment of character strings. The best user interface to FORTRAN based subroutines is probably based on character strings with a form of delimiter or end-of-string marker. As an example, suppose the routine TEXT displays a text string on a plot. Then, typically, the user would like to include a statement of the form

CALL TEXT(X,Y,"THIS IS A STRING")

in his program to cause the characters to appear on the plot.** Character strings are typically stored in packed form by a compiler. Since the graphics system needs to find the terminator (and perhaps other graphics control characters in the string), it needs to be able to access individual characters in the string. FORTRAN has no facility to access individual characters.

To do this, primitive routines can be defined which retrieve and store a single character in a string. Alternatively, primitive routines can pack and unpack entire character strings. Either of these approaches provides enough power for the necessary character manipulation. On any given machine it is normally trivial to implement these functions, either in assembly language or at times in non-portable FORTRAN.

The portable software package may also need to know something about its environment. To handle this, one other machine-dependent function can be assumed, which will provide information describing such characteristics of the local system as the number of characters that fit in an integer variable, the string terminator, and the output unit number. This technique is used in [PORT]. This information could also be supplied by a preprocessor.

2.2. Adaptability to Various Output Devices

To a large extent the device-to-device portability of graphics routines depends on the existence of a *hierarchical structure* of routines that make up the system. This concept is illustrated by Figure 1, which shows how high-level routines, such as those designed to produce scatter plots, etc., are built upon a succession of intermediate and low-level routines. Eventually all of the graphical operations carried out within the system are mapped into the operations of drawing lines, plotting points and text, and scaling. Scaling encompasses the operations necessary to transform the various plotting coordinates into physical coordinates needed by the plotting device.

Routines at this level are typical of those supported by vendor-supplied software for graphics terminals [Tek, HP]. At a level somewhat more primitive, operations are provided that deal in physical (device) coordinates, and perform such simple tasks as positioning the device at a specified point, plotting a single character at the current position, and drawing a line segment. These operations are characteristic of the actual hardware interface to graphics terminals.

When faced with the problem of interfacing a new graphics device to the portable system, it is only necessary to decide what level of routines is most like the operations required by the terminal. Primitive devices (line printers, Diablo terminals) operate at the lowest level, while newer microprocessor based devices often can handle some of the scaling operations, generation of text strings, etc.*

The critical point is that if the system is organized into a well defined set of routines that provide a number of levels of support, it is normally easy to construct a device interface. The operations that any new device can carry out usually each correspond to a routine at one of the levels of the system.

2.3. Operating System Considerations

Just because the code to support a particular device fits in well with the hierarchy mentioned in Section 2.2, it does not mean that overall device support is easy. Character codes can complicate matters. For example, on many

** A Hollerith constant could be used in place of the quoted string in this call. The termination character (') could be added by a preprocesso. Alternatively, the length of the character string could be given as another argument; however, since computers count better than people, the string terminator seems preferable.

* There are, in addition, some terminals with operations more complex than ordinarily required for most data analytic applications. A more comprehensive graphics package is required to adequately utilize hardware features such as dynamic rotation and 3-dimensional transformations [SIGGRAPH].

Hierarchical Structure of
Graphical Software

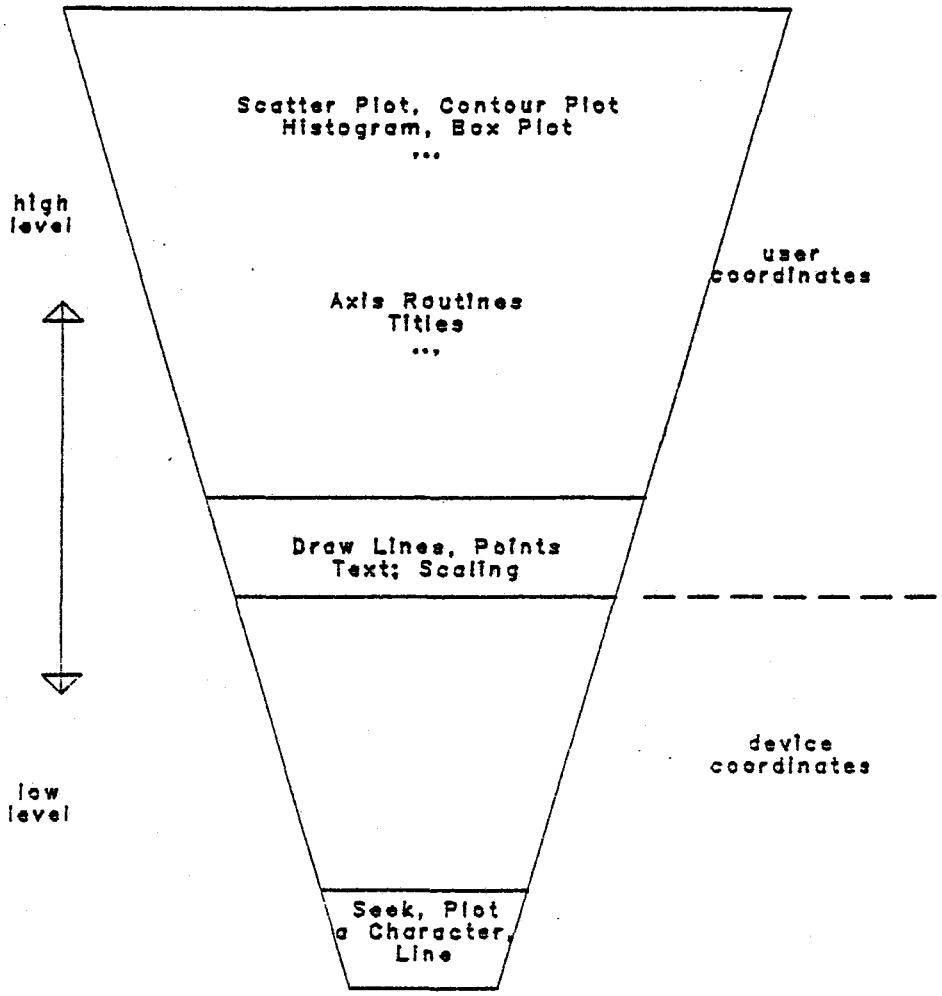


Figure 1

terminals, certain ASCII characters perform specific device-control functions, such as erasing the screen. It is easy for the software supporting the device to determine the appropriate ASCII characters to send to the terminal. However, operating systems often make it difficult to get the control characters to the terminal. For example, standard output routines may assume that any characters going to the terminal must be transliterated from the machine's own internal character set to ASCII, and there may be no internal characters that transliterate to ASCII control characters. Because of this, it is necessary to assume a primitive routine which will take a set of ASCII characters and get them transmitted to the device. This routine is normally easy to write for any particular installation.

Another operating system consideration that crops up early in the subroutine library approach concerns the exact way that libraries are created and searched on the target machine. One possible organization of a subroutine-based graphics system consists of one library for the device-independent code, and then a number of device libraries for the device-dependent routines. The local operating system can conspire against this approach in several ways.

- It may be necessary to organize the routines in a specific order on the libraries. Sequential libraries often assume that the routines are ordered so that any routine appears on the library before all routines that it calls. In a hierarchically structured set of routines this ordering can be determined, but it is inconvenient.
- It may be difficult or expensive to create libraries, e.g., on IBM systems, separate compile/link-edit steps are necessary for each routine on the library.

Either of these problems can mean the difference between the easy installation of the portable graphics system and installation that requires a substantial amount of work and programming talent.

Even more mundane, but still not easy to solve, are the problems of transmission of the original source of the system to the target machine. The conspiracy of character sets and tape formats often leads to relatively difficult operations on the receiving end.

3. Experience with GR-Z

GR-Z is the name of a portable set of algorithms designed at Bell Laboratories. The features of the system which make it especially suitable to graphics for data analysis have been described in [GR-Z]. Most of the considerations of portability mentioned earlier are reflected in the design of the GR-Z system.

The portability of GR-Z has been tested by installation on a number of different computing systems, including Honeywell 6000, IBM 370, CDC 6600, Harris 7, and DEC 10. Installation of GR-Z on a new machine typically takes one day, with the majority of that time occupied in initial reading of the source tape and creation of the primitive routines needed for the target machine.

GR-Z supports a number of different graphical devices, ranging from line printers to storage scopes and interactive pen plotters. The device driver code for the printer is written in portable FORTRAN. A number of off-line devices have been supported by connections to another low-level graphics system. Creation of a GR-Z device driver routine is generally easy, so that useful graphics can be produced in several hours.

A master copy of the source code for GR-Z is kept in a machine independent form. When a version of GR-Z is produced for a new machine, a special system generation procedure is used to produce code with character strings tailored to the specific characteristics of the target machine.

In addition to source code for a number of devices and the relatively large amount of device-independent code, there is also an extensive set of test routines which is supplied to test both GR-Z and the locally written primitive routines.

4. Conclusion

It is possible to design portable graphical software. Through the use of software tools to verify the portability of the source code, it is possible to provide routines that will compile correctly at first try on a new machine. A suitable hierarchical structure of routines can make it easy to support many different devices. Properly chosen primitive routines can isolate operating system dependencies. However, in spite of these considerations, the operating system can provide problems that are difficult and not soluble in general.

References

[ANSI] American National Standards Institute, *American National Standard FORTRAN*, New York, 1966.

[GR-Z] Richard A. Becker and John M. Chambers, "On Structure and Portability in Graphics for Data Analysis", *Proceedings of the Ninth Interface Symposium on Computer Science and Statistics*, April 1-2, 1976.

[HP] Hewlett-Packard, *HP-PLOT/21 Software User's Manual*, August 1977.

[Luehrmann] Arthur W. Luehrmann, "Graphic Statements in the BASIC Language", *Proceedings of the Ninth Interface Symposium on Computer Science and Statistics*, April 1-2, 1976.

[PORT] P. Fox, A. D. Hall, and N. L. Schryer, *The PORT Mathematical Subroutine Library*, Bell Laboratories, Murray Hill, NJ, 1976.

[Ryder] B. G. Ryder, "The PFORT Verifier", *Software Practice and Experience*, **4**, 359-377, 1974.

[SIGGRAPH] "Status Report of the Graphic Standards Planning Committee of ACM/SIGGRAPH", *Computer Graphics*, **11**,3 Fall 1977.

[Tek] Tektronix, Inc., *PLOT-10 Terminal Control System, System Manual*, Document No. 070-4113-00, 1975.

VISUAL AND COMPUTATIONAL CONSIDERATIONS IN SMOOTHING SCATTERPLOTS BY ROBUST LOCALLY WEIGHTED REGRESSION

William S. Cleveland

Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

The scatterplot is one of the most powerful and most used statistical tools. With only a small amount of additional effort, the visual information can be greatly increased by plotting another set of points whose purpose is to summarize some aspect of the scatterplot.

1. INTRODUCTION

Figure 1 shows a scatterplot of points (x_i,y_i), for $i = 1,...,n$, where $n = 50$. In Figure 2 the same scatterplot is summarized by another set of points $(x_i,\hat{y}_i)$, for $i = 1,...,n$, which are plotted by joining successive values by straight lines. The point $(x_i,\hat{y}_i)$ portrays the middle of the distribution of the variable on the vertical axis, Y, given the value of the variable on the horizontal axis, $X = x_i$. The formation of the new points will be referred to as "smoothing" the scatterplot. The point $(x_i,\hat{y}_i)$ is called the smooth at x_i and $\hat{y}_i$ is called the fitted value at x_i.

The example in Figure 1 was generated by taking $x_i = i$, for $i = 1,..., 50$ and

$$y_i = .02\ x_i + \epsilon_i\ ,$$

where the ϵ_i are a random sample from a normal distribution with mean 0 and variance 1. The linear effect is not easily perceived from the scatterplot alone, but is revealed when the smooth is superimposed.

In this paper we shall discuss a method for smoothing scatterplots called robust locally weighted regression. The details of the method are given in Section 2. Various visual considerations and alternative plotting procedures are discussed in Section 3 and computational matters are discussed in Section 4. References [1], [2], [3, p. 225], [4], [5, Chapters 8 and 9], and [6] describe other methods for smoothing scatterplots.

2. ROBUST LOCALLY WEIGHTED REGRESSION

The method of smoothing used in Figure 2, which is called robust locally weighted regression, is defined by the following sequence:

(1) Let

$$W(x) = (1-|x|^3)^3 I(x)$$

where $I(x) = 1$ if $|x| \leqslant 1$ and $I(x) = 0$ if $|x| > 1$. Let

$$B(x) = (1-x^2)^2 I(x)\ .$$

(2) For each i let h_i be the distance from x_i to the r-th nearest neighbor of x_i. That is h_i is the r-th smallest number among $|x_i-x_j|$, for $j = 1,...,n$. For $k = 1,...,n$ let

$$w_k(x_i) = W\left(h_i^{-1}(x_k-x_i)\right)\ .$$

(3) For each i compute $\hat{\beta}_0(x_i)$ and $\hat{\beta}_1(x_i)$, the intercept and slope respectively, of a linear regression of y_k on x_k using weighted least squares with weight $w_k(x_i)$ at (x_k,y_k). That is, $\hat{\beta}_0(x_i)$ and $\hat{\beta}_1(x_i)$ are the values of β_0 and β_1 which minimize

$$\sum_{k=1}^{n} w_k(x_i)\,(y_k-\beta_0-\beta_1 x_k)^2\ .$$

Let

$$\hat{y}_i = \hat{\beta}_0(x_i) + \hat{\beta}_1(x_i)x_i$$

be the fitted value of the line at x_i.

(4) Let

$$e_i = y_i - \hat{y}_i$$

be the residuals from the current fitted values. Let s be the median of the $|e_i|$. Define robustness weights by

$$\delta_k = B\left(\frac{e_i}{6s}\right)\ .$$

(5) Recompute $\hat{y}_i$ for each i by fitting a line using weighted least squares with weight $\delta_k w_k(x_i)$ at (x_k,y_k).

(6) Repeatedly carry out steps (4) and (5) a total of one, two, or three times or until convergence occurs. The

final $\hat{y}_i$ are robust locally weighted regression fitted values.

The weights $w_k(x_i)$ decrease as the distance of x_k from x_i increases. Thus points whose abscissas are close to x_i play a large role in the determination of $\hat{y}_i$ while points far away play little or no role. Increasing r, the number of nearest neighbors, tends to increase the smoothness of the smoothed points $(x_i,\hat{y}_i)$. Choosing r to be 20 to 80 of n should serve most purposes. A practical default value, used in the example of Figures 1 and 2 is $.5n$. The "tricube" weight function, $W(x)$, is used to weight neighbors since it results in certain desirable statistical properties [7].

The iterative fitting in steps (4) to (6) is carried out to achieve a robust smooth in which a small fraction of deviant points does not distort the results. Deviant points tend to have small robustness weights, δ_k, and therefore do not play a large role in the determination of the smoothed values. The "bisquare" weight function, $B(x)$, is used since other investigations have shown it to perform well for robust estimation of location [8] and for robust regression [9]. Two iterations of steps (4) and (5) are generally quite sufficient; this is the number used in the example of Figures 1 and 2.

3. VISUAL CONSIDERATIONS

3.1 Plotting the Smooth

The smoothed points can be plotted by joining successive points by straight lines as in Figure 2 or by symbols at the points $(x_i,\hat{y}_i)$. When the smooth is superimposed on the scatterplot the first method provides greater visual discrimination with the points of the scatterplot. But using lines raises the danger of an inappropriate interpolation. One possible approach is to use symbols initially when analyzing the data; then if a particular plot is needed for further use, such as presentation to others, the lines can be used if the initial plot indicates that linear interpolation would not lead to a distortion of the results.

The smoothed values also can be plotted on a separate grid with the same scales as the original scatterplot. This is particularly attractive for low resolution plots such as printer plots.

3.2 Symmetric Summaries

The method of summarizing the scatterplot in Section 2 is appropriate when Y is the response or dependent variable and X is the independent variable. In cases where neither variable can be designated as the response, the scatterplot can be summarized by plotting the smooth of Y given X and the smooth of X given Y.

3.3 Summarizing Scale and Choosing a Scale Stabilizing Transformation

The smoothed points in Figure 1 portray the location of the distribution of Y given $X = x_i$. It is often useful to have, in addition, a summary of the scale. This can be done by plotting $|y_i - \hat{y}_i|$ against x_i and computing and plotting a smooth, $(x_i,\hat{s}_i)$, of this scatterplot.

If the scale of y_i is a function, $\sigma(\mu)$, of the location of y_i then the transformation of y_i which stabilizes the scale [10, p. 425] is

$$t = \int \frac{1}{\sigma} .$$

Suppose t is a power transformation

$$t(\mu) = \begin{cases} \dfrac{\mu^p - 1}{p} & p \neq 0 \\ \log \mu & p = 0 \end{cases} .$$

Tukey [5, p. 103] has suggested a procedure for choosing a scale stabilizing power transformation for batches of numbers, which can be extended to choosing one for scatterplots. From the above equations we have

$$\log \sigma(\mu) = -\log t'(\mu) = -(p-1)\log \mu .$$

A plot of $\hat{s}_i$ vs. $\hat{y}_i$ describes the function $\sigma(\mu)$. Thus a plot of $\log \hat{s}_i$ vs. $-\log \hat{y}_i$ will, apart from sampling fluctuations, follow a line with slope $p-1$. Thus p can be chosen by fitting a line to the plot, either by eye or by some numerical method.

3.4 Judging the Amount of Smoothing

The most practical method for choosing r, the number of nearest neighbors, is to study the visual display. The objective is to choose r as large as possible without distorting the underlying pattern in the scatterplot.

The fitted value, $\hat{y}_i$, in step (3) of Section 2 can be written as

$$\hat{y}_i = \sum_{k=1}^{n} r_k(x_i) y_k ,$$

where $r_k(x_i)$ depends only on $x_1, \ldots, x_n$. The equivalent number of parameters

$$enp = 2 \sum_{i=1}^{n} r_i(x_i) - \sum_{i,k=1}^{n} r_k^2(x_i)$$

also can be used to judge the relative amounts of smoothing for different values of r. An interpretation of enp arises from considering the variability in the residuals, e_i, of the fitted values in (3). Suppose the y_i are independent and have common variance σ^2 then

$$enp = n - \frac{E \sum_{i=1}^{n} e_i^2}{\sigma^2} .$$

Suppose the e_i were the residuals from a linear least squares regression of y_i on x_i using q parameters, then enp would be equal to q. Thus enp for locally weighted regression can be interpreted as an equivalent number of parameters.

In [7] it is shown that enp can be approximated (when the weight function is tricube) by

$$2(1 + \frac{n}{r}) .$$

Thus for the default value of $\frac{r}{n} = .5$ described in Section 2, the approximate equivalent number of parameters is 6.

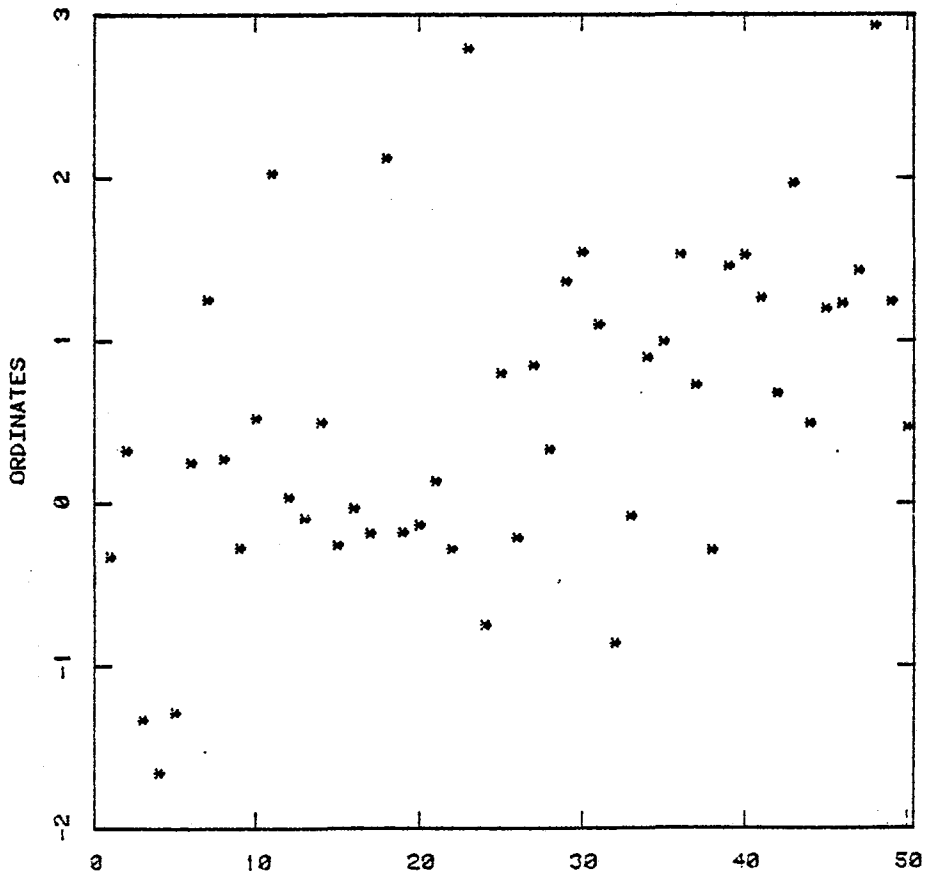


FIGURE 1

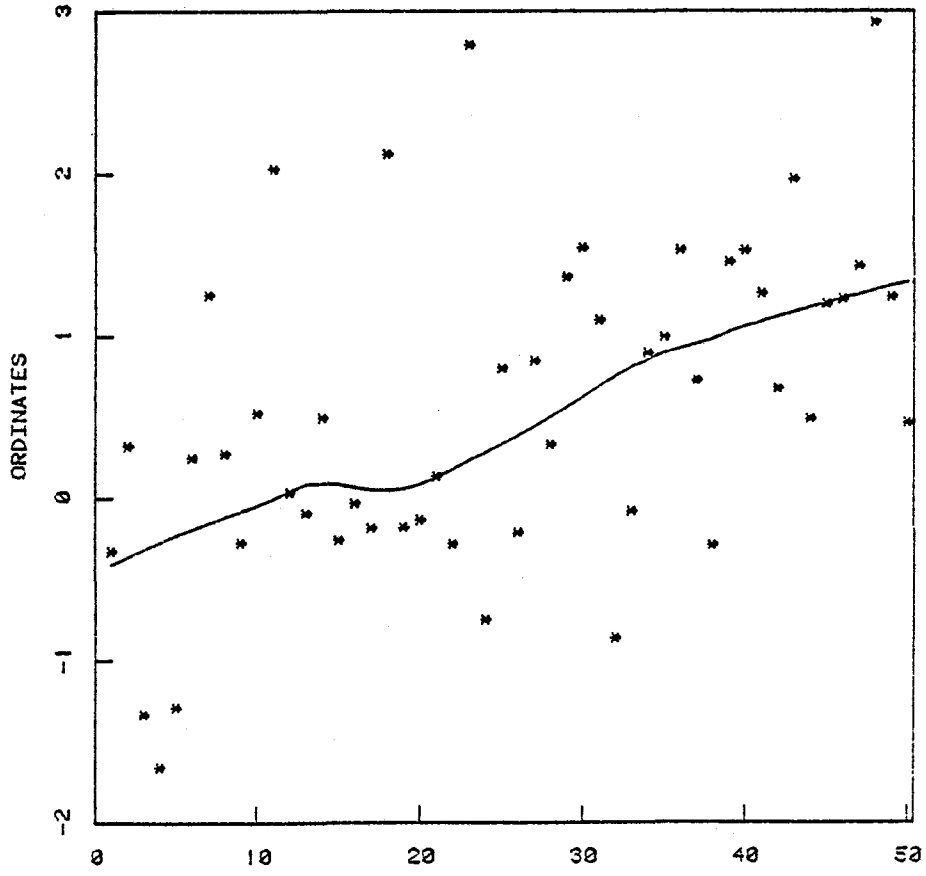


FIGURE 2

3.5 Nearest Neighbor vs. Equal Resolution

An alternative to choosing h_i through nearest neighbors is to use a constant value h for the computation of all fitted values. This provides equal resolution over all regions of the scatterplot but leads to appreciable increases in the variance at isolated points and at the ends of the scatterplot. The selection of the nearest neighbor routine is based on its more satisfactory performance, particularly at the ends of the scatterplot, for the applications which I have encountered. However, other users may find equal resolution more satisfactory in other applications.

4. COMPUTATIONAL CONSIDERATIONS

4.1 Reducing the Computations

Suppose the x_i are ordered from smallest to largest and let $x_{a(i)}, \ldots, x_{b(i)}$ be the ordered r nearest neighbors of x_i. The values of $a(i+1)$ and $b(i+1)$ can be found from $a(i)$ and $b(i)$ using the following scheme:

(1) Let $A = a(i)$ and $B = b(i)$.

(2) Let $d_A = x_{i+1} - x_A$ and $d_B = x_{B+1} - x_{i+1}$.

(4) a. If $d_A \leqslant d_B$ then $a(i+1) = A$ and $b(i+1) = B$.
b. If $d_A > d_B$ replace A by $A+1$ and B by $B+1$ and return to (2).

(4) h_{i+1} is the maximum of $x_{i+1} - x_A$ and $x_B - x_{i+1}$.

Thus this scheme can be used to save computations by computing the fitted values at x_1, then x_2, etc. Only $x_{a(i)}, \ldots, x_{b(i)}$ need be considered in the weighted least squares computation of $\hat{y}_i$ since $W(x) = 0$ for $|x| \geqslant 1$. This saving would not be achieved by using a weight function which becomes small but not zero for large x, such as the normal probability density.

4.2 Computation Time

An experiment was run to determine the run time of the smooth using the scheme in Section 4.1 for the nearest neighbor algorithm in Section 2 with one iteration. (The portable FORTRAN routine is available from the author.) Each additional iteration would increase the time by slightly less than 50 of the time required for one iteration. Consideration of the algorithm shows that the run time is independent of the configuration of the x_i. (This would not be true for the equal resolution algorithm.) The experiment was run for all 20 combinations of 5 values of $f = r/n$ (.2, .4, .6, .8, 1.0) and 4 values of n (25, 50, 100, and 200). A least squares fit to the log (base 10) run time (cpu milleseconds) resulted in the fitted equation

$$\log \textit{time} = -.49 + 1.98 \log n + .87 \log f .$$

The estimate of the residual standard error is the standard error of the log times is .718 log milleseconds. Thus the equation provides a very close fit to the data.

4.3 Thinning

The computations for the nearest neighbor algorithm are, as shown in the previous section, approximately of the order $f^{.9}\, n^2$. For scatterplots with fewer than 50 to 100 points the computations present no problems. Plots with more points generally need not incur the cost of using all the points since computing the smooth at a subset of the points will generally perform satisfactorily. (The smooth can, of course, still be superimposed on the full scatterplot.)

Two possible methods of thinning are to select every t-th value of the ordered x_i or to form a grid of equally spaced points on the horizontal axis and select, for each grid value, the x_i which is closest to the value.

4.4 Locally Weighted Regression of Order d

Steps (3) and (5) of the procedure in Section 2 can be generalized by fitting a polynomial of degree d, where d is a non-negative integer. Choosing $d = 1$ appears to strike a good balance between computational ease and the need for flexibility to reproduce patterns in the data. The case $d = 0$ is the simplest, computationally, but in the practical situation an assumption of local linearity seems to serve far better than an assumption of local constancy since the practice is to plot variables which are related to one another. For $d = 2$, however, computational considerations begin to override the need for having flexibility.

ACKNOWLEDGEMENT

The computing considerations for smoothing scatterplots were, in part, shaped by some very helpful discussions with my colleague, Rick Becker.

William S. Cleveland

REFERENCES

[1] Clark, R. M. (1977). "Non-parametric Estimation of a Smooth Regression Function," *Journal of the Royal Statistical Society,* Series B, 39, 107-113.

[2] Cleveland, W. S., and Kleiner, B. (1975). "A Graphical Technique for Enhancing Scatterplots with Moving Statistics," *Technometrics,* 17, 447-454.

[3] Ezekiel, M. (1941). *Methods of Correlation Analysis,* Second Edition, Wiley, New York.

[4] Stone, C. J. (1977). "Consistent Nonparametric Regression," *The Annals of Statistics,* 5, 595-620.

[5] Tukey, J. W. (1977). *Exploratory Data Analysis.* Addison-Wesley, Reading, Mass.

[6] Watson, G. S. (1964). "Smooth Regression Analysis," *Sankhya,* 26, Series A, 359-372.

[7] Cleveland, William S. (1977). "Locally Weighted Regression and Smoothing Scatterplots," (submitted for publication).

[8] Gross, A. M. (1976). "Confidence Interval Robustness with Long-Tailed Symmetric Distributions," *Journal of the American Statistical Association,* 71, 409-416.

[9] Gross, A. M. (1977). "Confidence Intervals for Bisquare Regression Estimates," *Journal of the American Statistical Association,* 72, 341-354.

[10] Kendall, M. G. and Stuart, A. S. (1977). *The Advanced Theory of Statistics,* Volume 1, 4th Edition. Hafner, New York.

COMPUTER-GENERATED MOVIES AS AN ANALYTIC TOOL

by

Raymond L. Elliott
Los Alamos Scientific Laboratory
Los Alamos, New Mexico 87545

ABSTRACT

One of the problems faced by the users of large, sophisticated modeling programs at the Los Alamos Scientific Laboratory (LASL) is the analysis of the results of their calculations. One of the more productive and frequently spectacular methods is the production of computer-generated movies. This paper presents an overview of the generation of computer movies at LASL. The hardware, software, and generation techniques are briefly discussed.

A number of programs at the Los Alamos Scientific Laboratory (LASL) are concerned with the modeling of physical phenomena as a function of time. The applications vary tremendously from the collision of atoms to the depletion of an oil reservoir; from the modeling of an ion accelerator to cooling of a metal mold; from the detonation of high explosives to the emergency cooling of a reactor vessel. These activities, plus many others, have several things in common. They all are modeled on large computers, are time dependent, and use computer graphics to interpret the results.

The people involved with these programs make extensive use of computers and are faced with the continuing problem of interpreting the results of their calculations. A common method is to generate volumes of computer listings. These are not readily assimilated and are of limited use (other than to serve notice to supervisors that people are producing results, i.e., something that can be touched). Another common method to is generate graphic plots of parameters. These can be on microfiche, microfilm, paper, or graphics terminals. For interpreting computer results, a picture is worth much more than a thousand numbers. Frequently, the user is able to understand his calculations with a few plots since a large amount of information is presented in a concise, easy-to-understand form. This is sufficient for many applications. For others, another technique is used (and, in many cases, required), i.e., the production of computer-generated movies, which typically show the modeling of physical phenomena as a function of time.

Computer-generated movies are used primarily in two ways. They are used as an analytic tool in the interpretation of computer calculations. They are also used to report results of the computer calculations to management and other interested people. These reports are typically much better received than a pile of paper. The content of the movie varies depending on its intended use. In general, movies used as analytic tools contain a large amount of information and must be viewed repeatedly to gather that information. A movie used to show results to others should be simple enough to be understood at a single showing.

Computer-generated movies are made by producing each frame separately. Any plotting utility may be used to specify the lines and symbols that are used to draw each frame. The computer program that generates the data for the movie writes the information for every line and symbol for every frame on a file or set of files. For a complex movie, the files may contain tens of millions of computer words. Files are processed on a microfilm recorder that reads the data from the file, draws the picture on a cathode-ray tube (CRT), and exposes the film. The movies are generally produced on 16-mm film, however, it is possible to generate 35-mm film. Color is produced by placing computer-controlled color filters in the optic path between the CRT and the film.

Computer-generated movies are expensive. They require significant amounts of computer time, microfilm recorder time, special

viewing equipment, and people time. Is it worth the cost? Most users find the benefits are substantial and are worth far more than the cost. Some have made the generation of movies an integral part of their research process.

There are many techniques currently in use to minimize the cost, while retaining all the desired information. One technique is to process only as many frames as needed to describe the problem. Many processes can be sufficiently described in about one or two hundred frames. At a projection rate of 16 frames/second, the movie will be 6 to 12 seconds long. However, this is insufficient viewing time to interpret the results. One solution to this problem is to generate multiple copies of each frame or to generate interpolated frames to make the movie any length desired. A far simpler and less expensive solution is to view the movie on a variable speed, reversible projector--slowing down to one frame per second if needed or stopping and reversing to examine an interesting segment of the movie. A variable speed, reversible projector is a wise investment if one is generating computer movies regularly.

Another technique used on some of the longer running production programs is the generation of a data file and postprocessing this file to make a movie. The file contains data sets saved at various times during the execution of the parent problem. Each data set contains those parameters the user needs to interpret his results. Thus, the file contains enough information to describe the complete computer calculation of a given problem. The problem times of the data sets need not be on a fixed interval. This enables one to save little data when the action is smooth and much data when the action is turbulent. The postprocessing movie program generates frames on a fixed interval using interpolation when needed. Many versions may be generated from the same data file using different problem parameters and plotting techniques, thus giving the user many ways to look at his data. Also, data from different computer runs may be combined to give problem comparisons. As a side benefit, the user can take advantage of an interactive graphics system to look at and modify pictures made from this data prior to making the movies.

Some users take advantage of another technique in their movie generation. They simply put as much information as possible on each frame. This results in movies with many distinct plots on separate areas of the frame. Each plot may become very complex. This naturally results in a movie that contains a very large amount of information and must be viewed many times to assimilate the information. This technique provides another way to correlate the interaction of many parameters during the course of a computer run.

Most computer-generated movies use color; the incremental cost over black and white is small. Color provides an easy way to present more information per frame. It is used to highlight areas of interest and to show more features, such as material differences. contour line identification, and problem conditions. The uses of color are as varied as the applications and have become an essential tool in those applications.

The Los Alamos Scientific Laboratory has a number of movies containing computer-generated segments that are used as reports on various activities. Most of these segments were initially generated as a part of the analysis of computer output. These movies demonstrate a variety of techniques for problem analysis. The films may be borrowed for short periods for educational, nonprofit, and noncommercial screening. Computer-generated movies such as these are playing an ever increasing role in the interpretation of the results of computer calculations and are becoming one of the more powerful tools available to scientists at the Los Alamos Scientific Laboratory.

REFERENCES

COMPUTER-GENERATED MOVIES

FILM-Y-264 THREE-DIMENSIONAL COMPUTER STUDIES. 1973

Color, sound, showing time 3 1/2 minutes.

This film demonstrates the use of computer-generated movies as an aid in the interpretation of computer calculations.

FILM-Y-269 PHYSICAL SIMULATIONS WITH COMPUTER COLOR GENERATIONS. 1975

Color, sound, showing time 6 minutes.

This film demonstrates the computer display of the same problem in three different ways: Mesh Plot, Contour Plot, and 3D Isometric Plot.

FILM-Y-271 SOLAR ARCHITECTURE. 1975

Color, sound, showing time 4 1/2 minutes.

This film demonstrates the use of three-dimensional computer graphics to achieve the best solar collector orientation during architectural planning. The National Security and Resources Study Center at Los Alamos is the example used.

FILM-Y-281 COMPUTER MOVIES AID TO ENERGY RESEARCH. 1975

Color, sound, showing time 10 minutes.

This film shows four areas of energy research where computer-generated motion pictures are used in design considerations. The four areas are: High Powered Laser Pulses, Solar Heating and Cooling, Laser Fusion Power Plants, and Pollution Studies of Power Plant Stacks.

FILM-Y-285 MATRICES AND THEIR SINGULAR VALUES. 1976

Color, sound, showing time 6 1/2 minutes.

Using computer graphics, a visual as well as oral explanation is made concerning the importance of matrices in the solution of certain types of problems. The actual solution procedure is discussed and visually demonstrated.

FILM-Y-360 THERMAL ANALYSIS IN MOLD DESIGN. 1977

Color, sound, showing time 4 minutes.

All of the visuals in this short film were produced by computer. These graphics represent the mathematical modeling of experiments in mold design. The developmental history of a specific mold is used to demonstrate the process.

To borrow these films write to:
Los Alamos Scientific Laboratory
Attention: Report Librarian
P.O. Box 1663, MS 364
Los Alamos, New Mexico 87545

Computer Graphics Standards and Statistical Data Plotting

James D. Foley
Department of Electrical Engineering and Computer Science
The George Washington University
Washington, D.C. 20052

Computer-generated plots are important for statistical data analysis, yet many plot programs cannot be shared because there are no generally-accepted graphics package standards. This paper discusses the advantages and disadvantages of graphics standardization, describes the various ways in which graphics standardization might be approached, and presents the proposed standard graphics subroutine package known as the Core System.

1. Introduction

Computer-generated graphs and plots have gained an important place in the world of statistics for use in data analysis and presentation. Over the years numerous computer programs have been developed for either passive plotting on output-oriented devices, or for interactive plotting and examination of statistics. In some cases the programs can be moved from one computer system to another and thus be shared, while in other cases this is not possible.

Broad, general sharing of statistics progams which use graphics requires at least three types of standardization:

- programming language standardization
- operating system standardization
- graphics standardization.

In this paper we focus on graphics standardization, with three purposes:

- to explore the advantages and disadvantages of graphics standardization.
- to describe the range of possible ways in which graphics standardization can be effected.
- to describe a proposed standard graphics subroutine package.

2. Advantages and Disadvantages of Standardization

Three advantages are generally ascribed to the standardization of computer graphics functions [1,2]. They are device independence, program portability (that is, machine independence), and programmer portability.

By device independence we mean the ability, at a particular computer installation, to use a graphics application interchangeably with several different plotting or display devices. This is of course beneficial for easing the replacement of old equipment with different new equipment, permits sharing of several disparate types of equipment. and simplifies the process of making hard copy of plots created interactively.

For interactive graphics applications, we extend the definition of device independence to include the interchangability of interaction devices such as keyboards, light pens, and tablets. This allows an application program to be used from interactive terminals with dissimilar interaction devices, or if a particular interaction device is not operational. Furthermore, this type of device independence expedites the human factors fine-tuning needed to optimize the man-machine interface for use with specific devices [3].

The most significant advantage of standardization is program portability, or machine independence. This is the ability to move a graphics application from one computer installation to another. Thus a program running on a PDP-10 with a GT-44 display might be moved to a CDC 7600 with an Imlac display. Doing this implies not only that the actual program be portable (hence written in a "standard" dialect of FORTRAN or other language), but implies graphics device independence as well.

The difficulties of achieving program portability are well known and are not to be minimized, even if an ANSI standard language is used. Similarly, graphics device independence is not without its problems. One must fully expect to invest effort in reprogramming some parts of the application. Indeed, in many circumstances a program can be called portable if

- the cost (in programmer effort, etc.) of moving the program is less than the cost of rewriting the program, and
- the required reprogramming is well-localized at clearly specified places in relatively few routines, and thus does not affect (or require

understanding) the program's structure and control flow.

The final advantage of standardization is the possibility of programmer portability, if the standardization is at the programming level (which usually implies a standard subroutine package). Programmers moving from the use of one graphics package to another are forced to learn new conventions, techniques, and concepts. Graphics programming is even worse in this regard than regular programming, because there is less commonality of terminology and concepts among graphics packages than there is among programming languages. A widely used standard graphics package would, in the long-term, eliminate much of this expensive and time-consuming relearning.

Against these many advantages must be weighed and balanced the disadvantages, which are very similar to the disadvantages of using standard programming languages. Just as do computers, plotters and interactive displays exhibit a wide range of architectural and performance characteristics. It is therefore difficult to optimize the use of a specific device with a standard general-purpose graphics system. (This is more of a problem with interactive displays than with plotters, which tend to have many generic similarities.)

If the standard does not include enough features, then there is the danger of not being able to effectively use some capabilities of high-performance displays such as those of Evans and Sutherland, Adage, and Vector General. On the other hand, if the standard includes too many features, software simulations will be required with less capable systems. This situation, shown in Figure 1, requires a compromise solution.

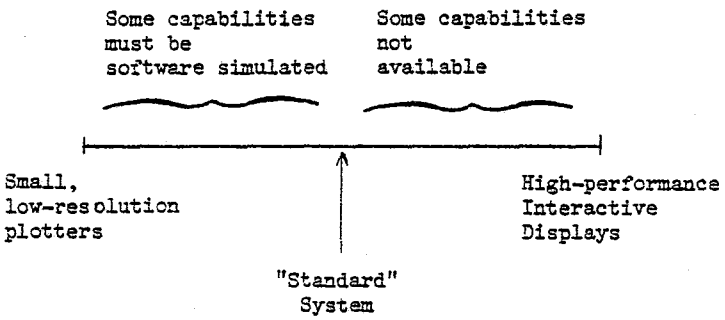


Figure 1

Another disadvantage attributed to the use of a graphics standard or any high-level graphics package is the loss of execution-time efficiency occasioned by removing the application programmer from the details of the display system. But this is exactly the argument once advanced against the use of FORTRAN and other high-level programming languages. Just as we have come to recognize the programmer efficiency of higher level programming languages, so too will happen with higher level graphics packages.

3. Approaches to Standardization

Figure 2 shows a useful view of how a complete graphics application system is organized. The "application program" itself is what gives any such system its individuality and uniqueness. All other parts of the system are in some sense "standard", typically existing prior to the time the application program itself is developed.

This then suggests that each interface between program units in Figure 2 has the potential for being formally (or informally) standardized. In the remainder of this section, we examine the possibility for standardizing at each of these levels, and report on what formal or informal standards exist at each level.

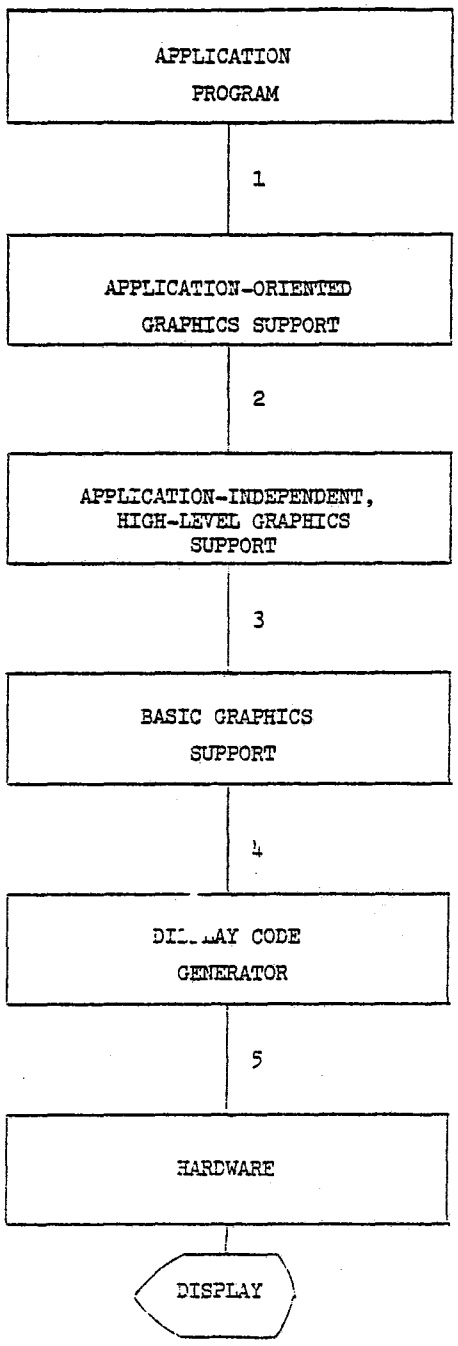


Figure 2

Interface Level 1

At interface 1, there exist various standard or commonly-used application-oriented systems. The statistical packages which do data plotting, even in simple forms such as scatter plots and regression curve fits, are at this level, as are various subroutine packages for statistical data plotting. Standardization at this level nicely addresses the needs of specific application areas, and has been successful with unsophisticated graphics devices such as line printers, plotters, and Tektronix displays because de facto standards exist at lower interface

levels for dealing with these devices (FORTRAN WRITE statements, CalComp subroutine calls for plotters, TCS/PLOT-10 for Tektronix). Achieving a wider degree of device-independence and portability at this level is in general quite difficult, unless standards exist at a lower level.

Interface Level 2

Interface level 2 is application-independent, but with a wide variety of rich capabilities. Some of the capabilities appeal more to certain application areas than to others. For instance, the DISSPLA subroutine package [4] has quite sophisticated plotting capabilities for producing graphics-arts quality charts (piecharts, barcharts, etc.), and can produce many different cartographic projections. On the other hand, the General Purpose Graphics System subroutine package [5] is oriented toward geometric manipulation and hierarchical modelling of objects and subobjects, and has very limited plotting capabilities. Both these packages are device- and machine- independent, as are the GINO-F [6] and GCS [7] subroutine packages. All four systems are widely-used, in some cases at well over one hundred locations, and therefore represent four different ad hoc "standards". Users of one of these "standards" can exchange programs with another user, but cross-standard program portability is difficult and expensive.

Why not adopt one of these four as a standard, or develop a new standard in the image of these systems? The problem is in the high level of the packages' capabilities in areas such as data plotting, cartographic projections, and hierarchical modelling. Obtaining the predecessor of standardization, a consensus of opinion for necessary features in each of these areas, appears difficult, if not impossible. Even if a consensus could be found, a single standard addressing these (and other) areas would be a monstrous system, prohibitively expensive to implement and use.

Interface Level 3

At interface level 3 we have a simpler, more fundamental set of graphics capabilities than at higher levels. The sorts of capabilities found here can usually be divided into seven basic categories:

1. Describing a 2D or 3D object for display in terms of simple output primitives, such as points, lines, and character strings, defined in application coordinates.
2. Controlling output primitives' attributes, such as color, intensity, line style, and text font.
3. Partitioning objects being displayed into segments, which form the units of modification in an interactive application. Segments can usually be created, deleted, renamed, and sometimes extended.
4. Controlling segment attributes, such as visibility and light-pen detectability.
5. Specifying the viewing position, viewing direction, field of view, and type of projection to be used in displaying objects in a viewport on a display surface.
6. Operator interaction using physical devices or their logical equivalent: PICK (light pen), LOCATOR (tablet), CHOICE (PFK), TEXT (keyboard), and VALUATOR (potentiometer).
7. Controlling the operational environment by specifying default values, selecting view surfaces, setting defaults and enabling or disabling clipping.

The proposed standard Core System subroutines package [1] is at this level, as are other subroutine packages such as Omnigraph [8]. The Network Graphics Protocol [9], developed for the ARPA computer network, is a data stream protocol rather than a subroutine package, but its semantics are at the basic graphics support level. Similarly, Bown and O'Brien's Graphical Task Instructions, or GTI's [10], define semantics and a data stream protocol. Of these two approaches (subroutine package, data protocol), the subroutine package is more attractive because it represents a more concrete and visible standard, and because it is easier to move a graphics program from one system to another if the standard interface is subroutine calls. Data stream protocols need to have the "syntactic sugar" of subroutine calls coated over the details of bits and bytes. Unless the subroutines are themselves standard, they must be moved with applications which use the data stream protocol.

This level of standardization is very attractive, because most of its capabilities will be used by many applications. Thus it is not too high-level. Similarly, many graphics applications can be implemented directly at this level, without the interposing levels shown in Figure 2. Thus it is not too low-level. However, higher-level graphics support can be built "on top" of the basic graphics support. The possibility of building on top suggests types of program portability as shown in Figure 3 (The style of this figure is attributable to James Michener).

In case (a), the application program interfaces directly to the Basic Graphics, and is portable in toto. In case (b), the application interfaces to High-Level Graphics through an interface which is standard within some sphere of endeavor. The application itself is movable to other computers which have the High-Level Graphics. If this is not so, then the High-Level Graphics can itself be moved. In case (c), which does not have a standard Basic Graphics interface, the High-Level Graphics cannot be easily moved. Case (d) represents a situation, quite likely to be found in practice, where the Application and High-Level Graphics must be moved together at the same time.

Interface Level 4

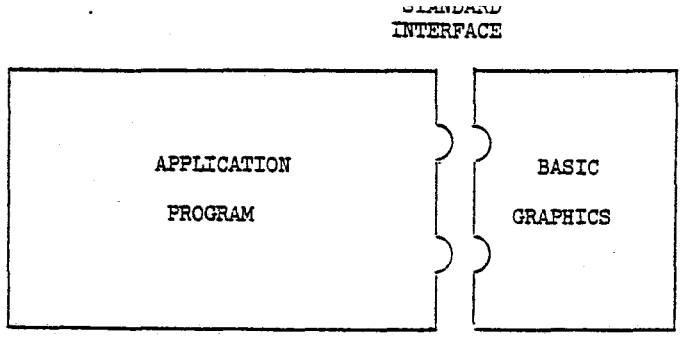
This level, often called the device driver level, is much more at a hardware-oriented level than the basic application -oriented level 3. It typically reflects the capabilities of specific display systems dealing with clipped and transformed 2D device coordinates, internal system names for segments, and the actual attributes and interaction devices supported by the display. Thus the device driver is no longer device-independent, as it typically does not provide extensive software support for unavailable hardware features. There are no standard device drivers for interactive graphics. DISSPLA, on the other hand, being an output-oriented system, has a very simple standard device driver interface - the basic command is "draw a line from here to there".

Interface Level 5

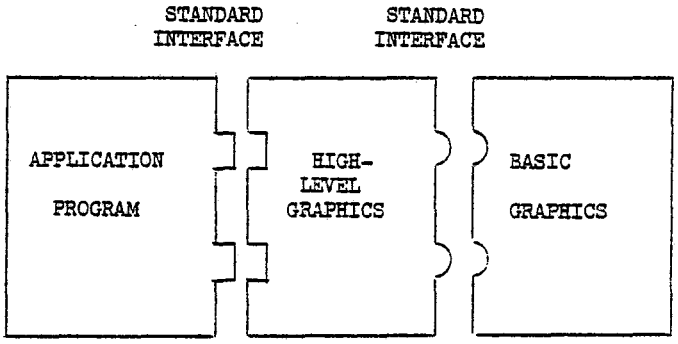
This is the hardware level - a standard here would define an instruction set for a display or plotter. There are not and never will be a standard at this level - manufacturers have (properly) shown no interest in this area. The most that will happen (and has happened in some cases) is that a microprocessor will be used to translate from one instruction set to another, much as some IBM computers can emulate older obsolete computers.

4. The Core System - A Proposed Standard

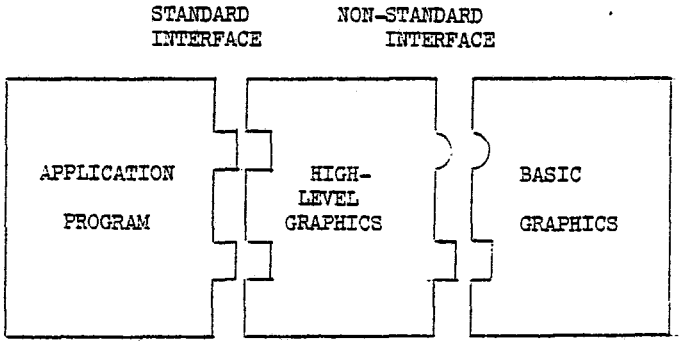
A standard subroutine package at interface level 3, Basic Graphics Support, has recently been designed by a subgroup of the Graphics Standards Planning Committee (GSPC) of ACM's Special Interest Group for



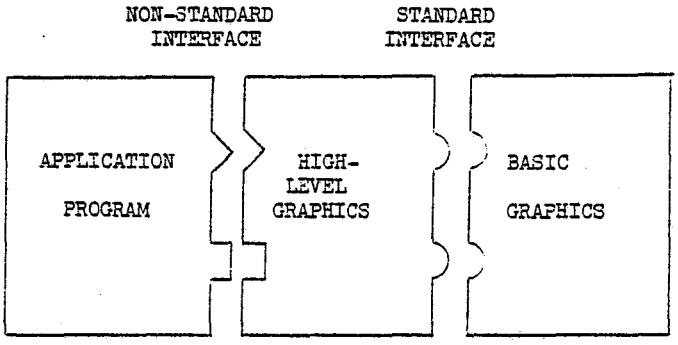
(a)



(b)



(c)



(d)

Figure 3

Graphics - SIGGRAPH. Published in the GSPC report [1] in August 1977, the package is already being implemented at several Universities and Government Laboratories, and is being studied by an American National Standards Institute (ANSI) Standards Planning and Recommendation Committee (SPARC). The SPARC, as well as national groups in Germany and The Netherlands and international ISO and IFIP subgroups, are using the Core System as a departure point for refinement and ultimate adoption as a national and international standard.

The Core System is oriented toward graphics devices ranging from simple plotters, through storage tube displays, up through medium-performance line-drawing interactive displays. Raster displays, currently in a rapidly changing state, were not included as a design target. However, they can be effectively used by adding a few additional facilities to the Core System.

The Core System's functional capabilities can be outlined according to the categories described earlier.

Graphics Output Primitives

Objects in 2D or 3D world (application) coordinates are described to the Core System as combinations of moves, lines, line sequences (called polylines), markers (a generalization of points), and text. A current position is used as the starting point for lines and text strings. The output primitives can be specified in absolute coordinates, or relative to the current position.

Output Primitive Attributes

Attributes affect the appearance of output primitives. Once an output primitive is created, its attributes can be changed only by deleting the primitive (using segments, described below), and then respecifying the primitive to the Core System. Primitive attributes are: linestyle (dot, dash, solid, etc.), linewidth, intensity, color, character font, character size, character spacing, character orientation, and character quality. This last attribute is used to indicate how closely the other character attributes must be followed. With low character quality, hardware character generators can always be used; with medium quality, sometimes; with high quality, nearly never.

Grouping, Naming, and Modification

All output primitives are placed into named segments which form the unit of modification. A segment can be created with two different statuses: with one, the segment is displayed once and no record is kept of the segment or its contents; with the other, the segment is retained until explicitly deleted. Such retained segments can be deleted, renamed, and have their attributes modified. In addition to the grouping provided by segments, output primitives within a segment may be placed in named groups. Segments can not contain references to other segments. The segment and group names are used for pick (light pen) identification.

Segment Attributes

Segments have dynamic attributes, which can be changed while a segment is being created or after it has been created. These attributes are visibility (whether the segment is actually displayed), detectability (whether the segment can be "seen" by a light pen or other pick device), and highlight (calling the operator's attention to the segment by blinking or intensifying it).

Viewing Transformations

Viewing of 2D objects is specified with the traditional viewport and window. The window may be inclined with respect to the principal axes. The object can be clipped against the window. In 3D, a viewer is imagined to be at an arbitrary point in three-space, looking toward any other point. The field of view is determined by specifying a rectangle in the projection plane, with the rectangle then mapped into the viewport. If the viewport is infinitely far away, an orthographic projection results; otherwise, a perspective projection results. The 3D object can be clipped against a 3D solid volume: a pyramid for perspective projections, and a solid rectangle for parallel projections. The image of a viewed object, be it 2D or 3D, can be translated, scaled, and rotated after the image has been created. 2D viewing is a proper subset of 3D viewing, with all 2D objects on the Z = 0 plane.

Interaction

Five classes of devices for operator input are supported. Picks provide the segment and group names of output primitives. Locators provide a position, while Valuators provide a scalar value. Text input devices provide character strings, and Choice devices provide a selection from several alternatives. The common physical prototypes for picks are light pens; for locators, joysticks and tablets; for valuators, dials; for text, alphanumeric keyboards; and for choice, programmed function keyboards. Each device class has a system-defined feedback, which can be turned on or off. Picks, text, and choice, when enabled, can place event reports on an input queue, which can be interrogated by the application program. Locators and valuators, when enabled, can be sampled by the application program, and can also have their input information placed on the input queue in conjunction with the occurrence of an event.

Control

The general operating environment of the Core System can be controlled by turning clipping on or off, selecting one of several display devices for output, setting the standard initial values for segment and output primitive attribute values, and establishing error handling procedures. Inquiry of Core System and device capabilities is also provided.

5. Levels of the Core System

Because the Core System is itself meant to be used in diverse environments with equipment ranging from plotters to highly-interactive refresh displays, four upward-compatible levels have been defined. An installation can choose the level best suited to its needs, thereby avoiding the space and/or time penalties which would accrue were unneeded capabilities included. The four levels in the Core System are called Basic, Buffered, Interactive, and Complete. They are also commonly called levels one to four.

Basic Level (1)

This simplest level is intended for "output only" applciations using plotters, microfilm units, or display terminals. Functions provided at this level are Viewing Transformations (except for transforming the image of a viewed object), Graphics Output Primitives and their Attributes, and Control. Segments must be used, but are displayed once and discarded. Since segments are not retained, their attributes cannot be changed, nor is it meaningful to delete or rename segments. No interactive capabilities are provided.

Buffered Level (2)

This second level is again intended for "output only" applications. The Buffered level allows selected segments to be retained from one output plot to another, allowing some segments to be used as fixed backgrounds on otherwise changing plots. The segment attributes of detectability and highlight are not provided, while visibility is. All operations on segments are meaningful. Named groups of primitives within a segment may be used, but are meaningless, and no interactive capabilities are provided.

Interactive Level (3)

This is the first level of the Core System which can utilize the full capabilities of DVST or refreshed displays to provide interactive capabilities. All functional capabilities for Grouping, Naming, Modification, and Interaction are provided. The only capability lacking is that of transforming the image of a viewed object.

Complete Level (4)

The outermost level of the Core System provides the complete set of functional capabilities, including transforming the image of a viewed object. This capability can be used to improve the effectiveness of many graphics satellite systems, and to access the dynamic transformation features of some high-performance display processors. Dragging an object with a pen or tablet is a simple example of an image transformation.

6. Summary

Graphics standardization, desirable for device independence, program portability, and programmer portability, can be effected in many ways. The Core System is meant to be a standard Basic Graphics system, usable directly by some applications and as the basis for higher-level or application-oriented support packages for other applications.

7. Acknowledgements

The Core System, which benefitted from critical reviews by several dozen members of the graphics community, was developed by the following committee:

Daniel Bergeron, University of New Hampshire
Peter Bono, Naval Underwater Systems Center
Ingrid Carlbom, Brown University
Timothy Dreisbach, SofTech, Inc.
James Foley, George Washington University
James Michener, Intermetrics, Inc.
Elaine Thomas Sonderregger, Information Sciences Institute, USC
Andries van Dam, Brown University.

The importance of the distinction between Basic Graphics and High-Level Graphics became clear at the IFIP W.G.5.2. Workshop on Graphics Methodology, held in April 1976. Had this distinction not been made, there might not be a Core System.

8. References

1. Status Report of the Graphics Standards Planning Committee of ACM/SIGGRAPH. Published as Computer Graphics 11 (3), Fall 1977.

2. van Dam, A. and Newman, W., "The Development of a Graphics Standard", To be published in Computing Surveys.

3. Foley, J. and Wallace, V., "The Art of Natural Graphic Man-Machine Conversation," Proceedings IEEE 62, 4 (April 1974), 462-470.

4. DISSPLA, Integrated Software Systems Company, San Diego, California.

5. Caruthers, L., van der Bos, J., van Dam, A., "A Device-Independent General Purpose Graphic System for Stand-Alone and Satellite Graphics", Proceedings of SIGGRAPH '77, published in Computer Graphics 11, 2 (Summer 1977), 112-119.

6. GINO-F Reference Manual, Computer-Aided Design Centre, Cambridge, England.

7. Puk, R.F., The 3D Graphics Compatibility System, U.S. Army Corps of Engineers, Waterways Experiment Station, Vicksburg, Miss., 1976.

8. Sproull, R.F., Omnigraph: Simple Terminal-Independent Graphics Software, Xerox Palo Alto Research Center, CS. 73-4, December 1973.

9. Sproull, R.F. and E.L. Thomas, "A Network Graphics Protocol", Computer Graphics 8, 3 (Fall 1974), 27-51.

10. Bown, H.G., C.D. O'Brien, R.E. Warburton, G.W. Thorgeirson, "System Independence for Interactive Computer Graphics Application Programs", Proc 4th Man-Computer Communications Conference, Ottawa, Canada, 1975, pp. 14.1 - 14.12.

Using a Low-Cost Digitizer to Capture Questionnaire Responses

J. Michael Hewitt
Chief, Computer Methods Laboratory
International Statistical Programs Center
Bureau of the Census
Washington, D. C. 20233

This paper discusses the use of a low-cost digitizer as a data entry device for statistical questionnaires. A possible production device is described, and its programming and use are discussed. Certain advantages gained by using a digitzer for data entry are cited.

Throughout the world today the prevailing mode of data entry for questionnaire responses is a key device, whether cardpunch, key to tape or key to disk. Keying questionnaire responses presents an array of difficulties which the data processor would like to overcome. Keying requires trained and increasingly expensive operators; it is highly prone to human error; it is relatively slow; finally, in most applications, it must be followed by key verification of the data.

Past attempts to eliminate the keying of questionnaire responses by using direct optical or magnetic mark-sensing of the questionnaire usually either have failed or have met with only qualified success. Typical problems have been related to the physical condition of the document and the difficulty of passing it through the document transport equipment. Paper reflectivity, size control, printing quality, unreadable questionnaire formats, and the care required when marking answers, or changing answers, also contribute to the difficulties with these mark-sensing techniques. All such problems can contribute to creating a document rejection rate of 10 to 20 percent, an unacceptable level particularly when dealing with large volumes of questionnaires.

The recent announcement of digitizer equipment at very moderate cost[1] -- purchase price of less than $500 per unit on quantity orders -- prompts speculation that the digitizer could be the basis for a new approach to statistical data entry, a tool which is simultaneously cost-effective, requires less-skilled labor, can achieve gains in speed of data entry, and can reduce human error by incorporating some visual verification into the data entry device.

A digitizer is a device used to convert material which is provided in graphic form into numerical data suitable for computer processing. It does this by referring to a specific set of two-dimensional coordinates assigned to every point on its "active surface." The digitizer is able to determine the coordinate pair defining the current location of its pen-like "stylus."

The technology actually used to determine the location of the stylus varies among different digitizer manufacturers. It appears that the particular technology is not important to the theoretical application described in this paper, other than that some methods may be more costly than others. It should be noted that this application seems to make very little demand on the digitizer in terms of the quality ranking factors of resolution, environmental stability, repeatability, slew rate, accuracy, and sampling rate.[2] Physical durability and general reliability and ruggedness will probably be more important. For example, it is possible that stylus wear may well be a problem.[3]

There is little doubt that a digitizer can be used for transferring statistical information from the questionnaire to computer-readable form. Similar types of data are being entered in conjunction with

the use of the digitizer as a graphical data entry tool -- often this is referred to as "menu" data entry. Known major applications using the digitizer in this manner include capturing real property information (along with the outline of the piece of property)[4] and preparing insurance proposals.[5] A questionnaire makes a particularly attractive target for the menu technique because the range of possible responses is usually limited. Relationship, sex, age or year of birth, and marital status are typical examples from a demographic survey questionnaire.

In addition, the digitizer offers a new technique for capturing responses to questions that have a continuous, rather than discrete, set of responses. Typically, in an opinion survey, a respondent might be asked to mark the place along a "response line" where an opinion is to be recorded. In a taste test the responses might range from "too sweet" at one end of the line to "too sour" at the other.[6] The digitizer device can be used to directly record the position that is marked on the line, and this can be expressed in some proportional value relevant to the item measured. A wider usage of this type of response line may result in easier-to-use forms and more accurate determinations of the real situation.

Because of a lack of experience with the digitizer, a number of questions need to be resolved before the device can see wide application as a data entry tool. Among these are: (a) whether a complete production data entry device can be constructed at a price competitive with other data entry tools (less than $1200, exclusive of developmental costs, in a minimum quantity of 100), (b) whether an unskilled operator can be trained to a reasonable level of production speed and accuracy in one day, or less, and (c) whether an operator can maintain speed and accuracy throughout a workday. A speed of one stroke per second on a well-designed form should be satisfactory, since some digitizer "touches" can be worth a number of keystrokes. A current user claims to exceed this rate by a factor of 2 to 3.

A complete production data entry device involving a digitizer can be projected to include five functional elements:

1. The digitizer board, stylus, cables, and associated electronics. Certain points on the digitizer board, outside of the form area, will be preprinted with a legend, and will have standard predefined values. These will include the digits 0 to 9 and certain necessary control functions.

2. An 8 bit microprocessor with sufficient RAM, ROM, I/O ports, and programming capability to manage the digitizer, provide some interactive dialogue with the operator, and save the data on some external device.

3. A digital cassette drive to save the input data for later processing by the mainframe computer. This device may also be used for passing special input programming to the microprocessor.

4. A tone generator to give the operator an audible signal that the digitizer input stroke has been detected by the device. This element may also give an audible signal when an input error has been detected. The tone could be transmitted through a small earphone so that it provides a private notification to the operator.

5. Some type of display panel, preferably alphanumeric. However, cost considerations may force a 7-segment numeric format. If alphanumeric, this should be 6-8 positions in length, enough for a brief message. If numeric, 3 digits should suffice. This display will be used by the microprocessor to communicate with the operator.

In addition, a conversion device will be required to transfer data from the cassettes prepared by the data input stations to a medium readable by the mainframe computer system. The conversion system could also be used to prepare special software for the input devices, if properly configured.

The basic conversion unit would contain at least two digital cassette drives, a microprocessor with communications display, and the conversion peripheral unit. A standard industry-compatible tape drive is a possibility, but the more likely interface would be an RS-232 interconnection, with the conversion unit appearing as a very high-speed modem to the target computer. Of course, other schemes are possible, but the RS-232 version is probably the least expensive in hardware costs, if not the fastest. Note that the conversion hardware may be an add-on that could be inserted into any of the data input units.

For a digitizer to substitute for a keyboard in a general data entry device, the intelligence of the microprocessor is required. The microprocessor is used to associate the coordinate location of the digitizer stylus with the proper ASCII code for the reponse.

If every different questionnaire were to require a special program written by a programmer the device would never be a viable substitute for the key entry of data. We can, however, envision a generalized program that will solve this problem. Suppose an operator is to enter data from a pre-coded questionnaire, responses having been marked on the questionnaire form by checking the boxes associated with response categories. A lead data-conversion operator can first "program" the hypothetical data entry device by placing the unit in a "teach" mode.

The operator might begin by first touching the stylus to four reference points at the corners of the questionnaire, thus defining the rectangle within which information will

be found. For example, suppose a particular question on the page has the allowable responses: 0, 1 and 2. While in teach mode, the operator uses the digitizer to program the microprocessor with the following operations:

1. The operator touches the premarked points on the active surface that designate the question number.

2. The operator then touches the premarked control point that is designated "enter question number".

3. The operator touches the box that corresponds to the answer for the "0" response.

4. The proper ASCII code related to these coordinates is then signaled to the device by touching the premarked point on the active surface labeled 0.

5. Since the corresponding response has only one digit, the end of the digit entries is signaled by touching the premarked point labeled "enter response".

6. This operation (steps 3 to 5 above) is continued for the responses of "1" and "2".

7. The operator then touches a control point marked "terminate question" to go on to the next question.

A similar series of actions would be used in teaching the device the length, orientation, and relative values to use when a question is associated with a response line.

Proceeding with these operations throughout the questionnaire page, the operator has now loaded the equivalent of a keypunch "program card" into the microprocessor. This program can now be saved or reproduced on the digital cassette drive for use at another session, or use by additional operators.

To actually record the data on this hypothetical data entry device the operator might go through the following steps:

1. The operator sits down at the work station at the beginning of the day and turns on the device.

2. In some manner (button or digitizer input) the operator tells the device that the equivalent of a keypunch "program card" is to be loaded. A cassette is put into the reader and the device reads the instructions for the particular form to be processed. The operator then removes this "programming cassette."

3. A data output cassette is placed in the cassette drive, and the operator begins to enter data. After the questionnaire is placed on the digitizer active surface, the first operation might be to touch certain preprinted points at 3 or 4 corners of the input document. This tells the input device how the printed portion of the form is oriented. The input device can then automatically adjust the subsequent coordinates to take account of: the sheet being skewed on the active surface, the sheet being square on the surface but having poor alignment in printing, or size variations in the form occurring after printing. An important advantage to note here is that as long as the form is readable, the darkness or density of printing is unimportant. In addition, marking the form will be very easy for the respondent or interviewer, since a check mark or "X" will suffice.

4. The fixed information for a batch will automatically be continued from document to document. The operator actually enters the data by touching the stylus to the marked response points. This can be called a "re-marking" mode of data entry. It should be emphasized that the device will save on the cassette ASCII codes that have been "taught" to it, rather than merely digitizer coordinates.

5. By touching appropriate predetermined points on the digitizer active surface, the operator can "backspace response," "backspace question," or "restart form." The input device does not write the data on the cassette until the form is fully input, so operator-detected conversion errors may be easily corrected and re-entered. The microprocessor code will (a) detect an error, (b) issue an audible alarm, and (c) display a message or error code when the operator touches the stylus to an answer out of question number sequence, touches an unpermitted point on the form, skips a question that requires a response, makes more than one response to a sub-question, or takes other improper action. The improper actions will be determined by the "edit input program" initially read in.

The above sequence of operations may at first appear to offer little advantage over keying, but a closer examination suggests otherwise. In addition to eliminating range errors, the programmed digitizer will not make column-shift errors. Each point on the active surface is fixed in unique association with a particular question response. A single point on the questionnaire surface may be associated with a multi-stroke response. One touch to one point on the surface may, for example, enter the four-stroke year "1977," or an alphanumeric response. In addition, the digitizer does not require the operator to move in linear progression across or down a page. If a right-handed operator begins entry at the bottom of the page and moves up the page with the stylus, the

operator's eye can travel ahead of the hand and thus increase operator speed.

Other aspects of this projected device represent considerable advantages over existing methods of data entry, such as keying or the various types of optical data input:

- If the industry-standard 8080A microprocessor is used, both COBOL and FORTRAN are available for development of more sophisticated input and editing software. In fact, it is likely that the CONCOR generalized editing system can be implemented in some form on the data input microprocessor. This would be an excellent base for the development of some interactive data cleaning software.

- The interviewer can easily change an answer by simply circling the wrong answer and marking the correct one. The digitizer operator touches only the uncircled answers.

- The physical condition of the forms as received from the field is not an important factor in capturing the data.

- Special scanning forms with tight tolerances, printed only by certain companies, are not necessary.

- If the data input step is combined with simple manual editing or simple coding, an entire step in handling the forms may be eliminated. (This is not recommended, however, when lengthy or complex coding is required.)

- In order to realize cost savings with increased speed and accuracy, the digitizer method does not substitute large capital expenditures for the employment of people.

If data can be digitized at least as quickly as it can be keyed, and if the hardware costs are roughly equivalent, the increased accuracy possible with the device is likely to give the digitizer widespread usage for converting statistical data. If in fact there are speed advantages, or smaller hardware costs, the digitizer will likely constitute a major step forward for data entry from statistical questionnaires.

The speculative nature of this paper should be stressed in that neither the projected device nor the required program code exists or is currently under development at the Census Bureau in January 1978.

REFERENCES

[1] Summagraphics Corporation, Fairfield, Connecticut. Called the Bit Pad, this device has an active surface of 11 inches square and 200 points per inch resolution.

[2] Digitizer Terminology and Comparability, Science Accessories Corporation, Southport, Connecticut

[3] Computer Systems and Services of Springfield, Springfield, Ohio. Private communication with the author.

[4] Computer Systems and Services of Springfield, cited in an Application Note by Summagraphics Corporation

[5] Phoenix Mutual Life Insurance Company, Hartford, Connecticut, cited in GRAF/PEN Notes by Science Accessories Corporation

[6] Pillsbury Corporation, Minneapolis, Minnesota. Telephone conversation between the author and Dr. Fred McCarron of Pillsbury. Pillsbury has constructed two devices similar to the above design, involving a digitizer, a microprocessor, and a digital cassette drive. These are used in conjunction with their taste testing operations.

A number of companies manufacture digitizers suitable to this task in addition to those referenced above.

COMPUTER GRAPHICS FOR EXTRACTING INFORMATION FROM DATA

by

RONALD K. LOHRDING, MYRLE M. JOHNSON, DAVID E. WHITEMAN

Energy Systems and Statistics
Los Alamos Scientific Laboratory
Los Alamos, NM 87545

Abstract

This paper presents computer graphics which are useful for displaying and analyzing data. Many classical and several newly developed graphical techniques in statistical data analysis are presented for small univariate and multivariate data sets. These include histograms, empirical density functions, pie charts, contour plots, discriminant analysis, cluster analysis, Chernoff "faces", Andrews' sine curves, three-dimensional plots, and probability plots.

Recent advances in data collection technology and computer data base management systems have made it imperative to utilize computer graphics for large data sets. Several innovative graphical techniques are presented to handle this situation.

Spatial relationships among the data (particularly geographic data) are difficult to visualize. Several cartographic techniques are presented which enhance the understanding of these spatial relationships within the data.

I. INTRODUCTION

The Energy Systems and Statistics Group at the Los Alamos Scientific Laboratory (LASL) is involved in several projects with energy-related data. Some of these projects involve small univariate or multivariate data sets, while others involve large data sets which require data management systems for efficient data manipulation. A statistically-oriented graphics package is presently under development; numerous modules have been completed. The purpose of this package is to provide graphical techniques for the initial examination of the data. This paper uses data from several projects to demonstrate some of these techniques.

In Section 2, we discuss graphical methods useful for a preliminary analysis of small data sets. In Section 3, graphical techniques which are appropriate for large data sets are presented. Finally, spatial relationships in geographic data sets are explored in Section 4. Throughout this paper, examples of computer graphics are used to illustrate the techniques. (The 35-mm color slides of computer-generated graphics shown at the conference are reproduced in black and white for this paper.)

II. PRELIMINARY DATA ANALYSIS OF SMALL UNIVARIATE AND MULTIVARIATE DATA SETS

Computer graphics for a preliminary raw data analysis may include histograms, empirical distribution function plots and probability plots. The data used in this section was collected on 17 variables for each of the 50 states plus the District of Columbia. The variables and their means and standard deviations are listed in Table I. Of particular interest is the variable household BTU consumption per

TABLE I

	VARIABLE		MEAN	STANDARD DEVIATION
1.	HHBTU	Household BTU per capita (10^4)	87.33	21.30
2.	DEGD	Heating degree day loads $\frac{\sum_{365} 65-Y\ {}^{\circ}F}{365}$ (10^3 °F) where $Y = \begin{cases} \text{average daily temp. if } Y \le 65^{\circ}F \\ 65^{\circ} \text{ if } Y > 65^{\circ}F \end{cases}$	5.00	2.23
3.	MAXT	Normal July maximum temperature (°F)	86.41	5.96
4.	PCAIR	Percent of households with air conditioning	33.73	18.44
5.	POP	1971 population (10^4)	4.04	4.36
6.	FRZR	Percent of population with freezers	32.90	10.94
7.	ONEP	Single individuals per housing unit	218.63	271.30
8.	PCURB	Percent urban population	66.47	15.11
9.	COML	Percent commercial sector $\frac{\text{commercial}}{\text{residential \& commercial}}$	36.71	3.31
10.	MEDIN	Median income (10^3)	9.17	1.45
11.	LOWIN	Percent of family incomes below gov't poverty levels	11.67	5.18
12.	SINGLE	Percent of single family houses	71.72	11.19
13.	NEWHS	Percent of houses built since 1960	25.91	7.92
14.	OLDHS	Percent of houses built before 1950	53.42	12.34
15.	AVEIN	Average income per capita (10^3)	3.96	.63
16.	LAT	Latitude of center of the state	39.48	6.44
17.	LONG	Longitude of center of the state	93.59	19.50

capita (HHBTU). The histogram in Fig. 1 suggests that the assumption of normality may be questionable. Two graphical tests of normality are shown in Figures 2 and 3. One test uses Lilliefors' test statistic;[13] the other uses a test statistic developed by Lohrding.[13] In the former, the normality hypothesis is tested by placing $(1-\alpha)100\%$ confidence bounds on the empirical distribution function (edf). The normal cumulative distribution function (cdf) with mean and variance estimated by the sample mean and sample variance is plotted. If the cdf falls outside the bounds placed on the edf, the hypothesis of normality is rejected at the α level of significance. In the latter test, the normality hypothesis is tested by placing $(1-\alpha)100\%$ confidence bounds on the normal cdf with mean and variance estimated by the sample mean and sample variance. If the edf falls outside the bounds placed on the cdf, the hypothesis of normality is rejected at α level of significance. In neither test is normality rejected at the 5% level of significance. A normal probability plot and a lognormal probability plot, two additional graphical techniques which may give further insight to the structure of the data, are given in Figures 4 and 5.

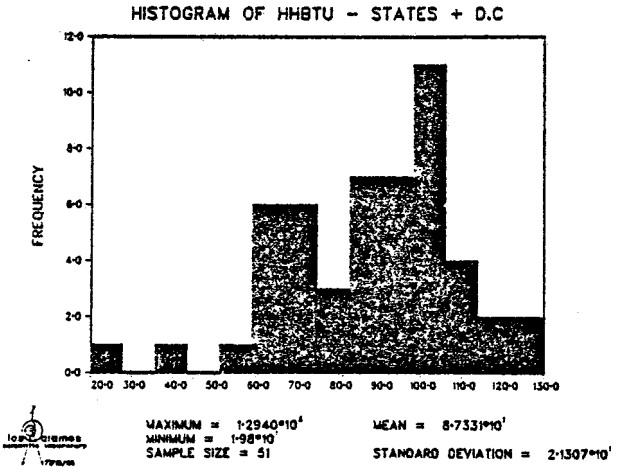


Figure 1

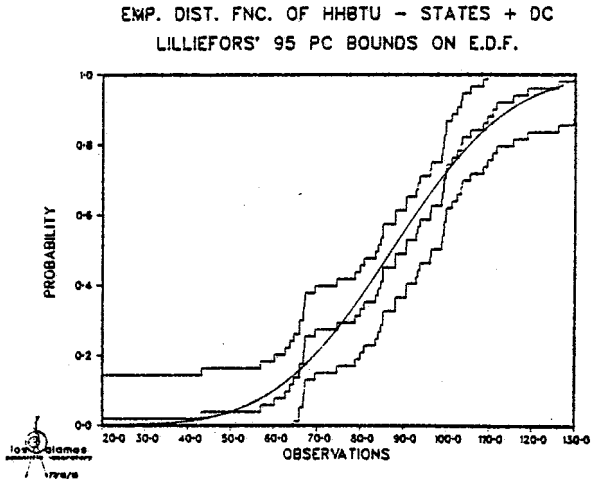


Figure 2

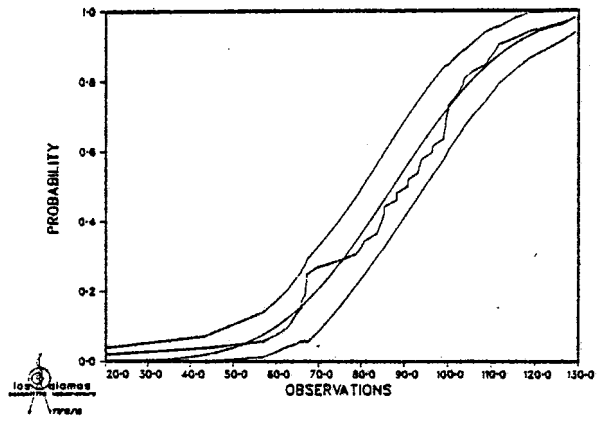


Figure 3

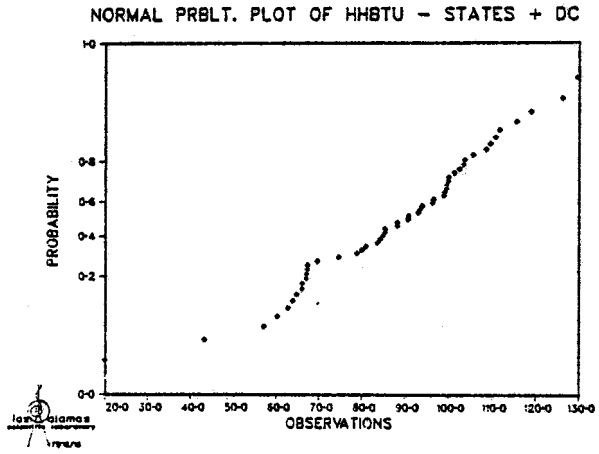


Figure 4

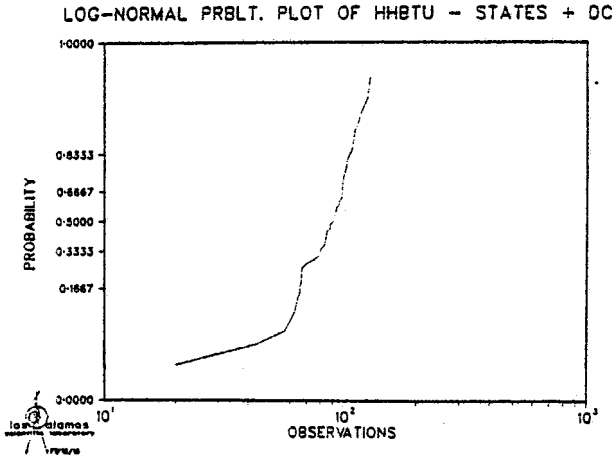


Figure 5

To describe the joint relationship of HHBTU to the 26 other variables (including transformations of some of the variables), a linear multiple stepwise regression procedure is used. Seventy-five percent of the variance is accounted for by two variables--degree days (DEGD) and percent urban population (PCURB). The equation of the fitted linear multiple regression model is

$$Y_i = 22.155 + 8.657\ X_{2,i} + 0.328\ X_{8,i}$$

where $i = 1,2,\ldots,51$

Y_i = HHBTU for the i th state (z axis)

$X_{2,i}$ = DEGD for the i th state (x axis)

$X_{8,i}$ = PCURB for the i th state (y axis).

Figure 6 shows a three dimensional graphical representation where the fitted plane and the data points are plotted. Lines are drawn from the data points to the surface to give some indication of the deviations.

In a nonlinear regression analysis, the equation of the fitted model is

$$Y_i = 33.835\ 77.607\left(\frac{X_{2,i}^2}{X_{3,i}}\right) + 1374.90\left(\frac{X_{2,i}}{X_{3,i}}\right)$$

where $i = 1,2,\ldots,51$

Y_i = HHBTU for the i th state (z axis)

$X_{2,i}$ = DEGD for the i th state (x axis)

$X_{3,i}$ = MAXT (maximum temperature) for the i th state (y axis).

The fit of the data to the surface is shown in Figure 7. The two largest deviations are Alaska and Hawaii.

Several techniques are available for displaying multivariate data. We first discuss a gray-level coded correlation matrix which displays the pairwise correlations between variables. The gray level scale ranges from plus one to minus one. Frequently, such a display is useful in directing attention to interesting relationships among the variables.

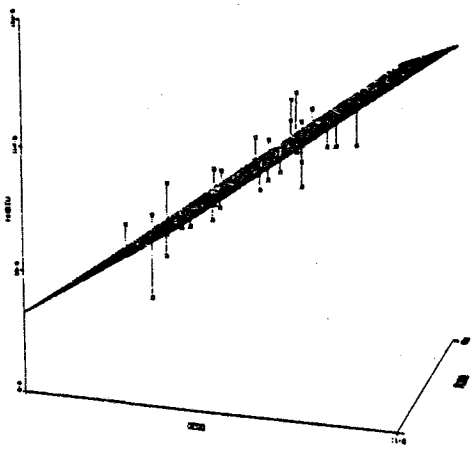


Figure 6

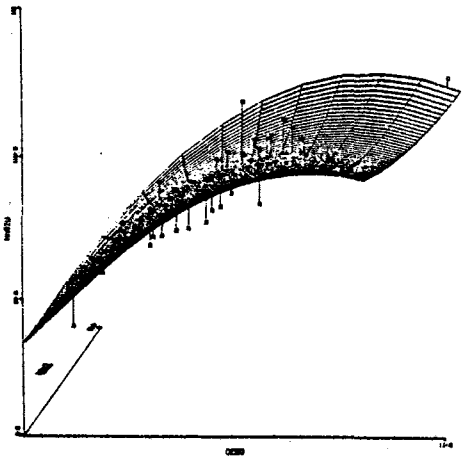


Figure 7

In Figure 8, note that HHBTU is positively correlated with DEGD, LAT, OLDHS, MEDIN, and AVEIN, negatively correlated with MAXT, PCAIR, LOWIN, SINGLE, and NEWHS, and not correlated with POP, FRZR, ONEP, PCURB, COML and LONG.

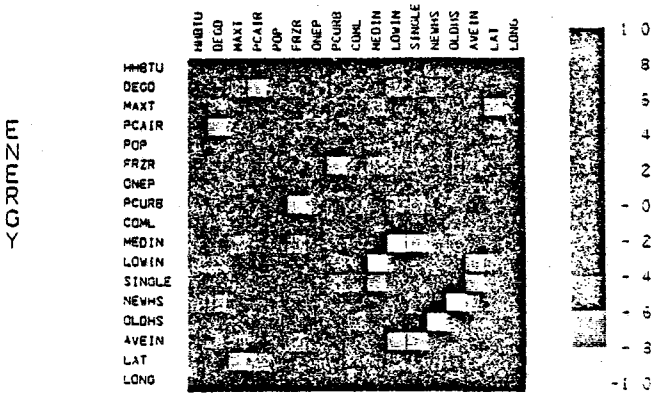


Figure 8

Another technique, called Andrews' sine curves,[1] displays multivariate data in a univariate plot. For each state, the 17 standardized variables are mapped into a trigonometric function on the range $-\pi$ to π. All 51 functions are then displayed simultaneously. Functions which form a relatively tight band indicate a cluster of states. When the original data is used, it is very difficult to detect any clusters as shown in Figure 9. However, if a principal components analysis is performed on the original data and the transformed data mapped into the same trigonometric function, it is easier to detect some clusters as shown in Figure 10.

A well-known technique in analyzing multivariate data is principal component analysis. As pointed out in Ref. 9, the first few and last few principal components are usually the ones of primary interest. Figure 11 is the plot of component 1 versus component 3. This plot reveals that Alaska, Hawaii, California, and New York are possibly aberrant observations.

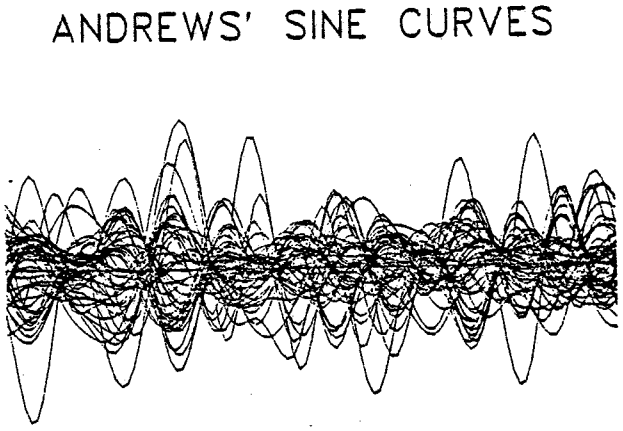


Figure 9

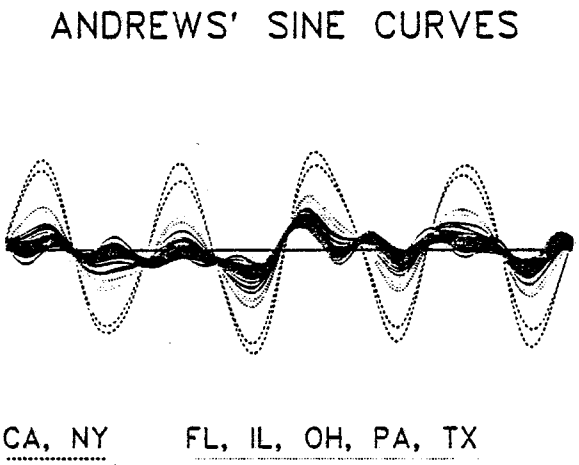


Figure 10

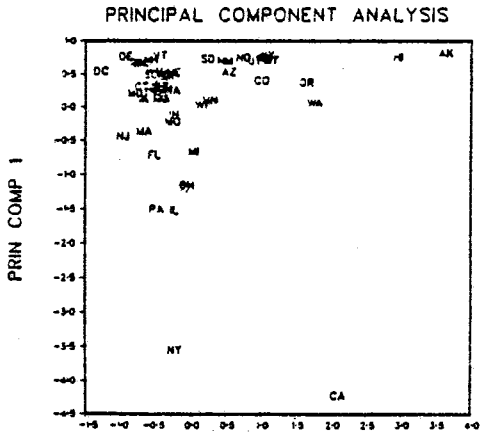


Figure 11

Another use of principal component analysis in locating possibly anomalous points is presented.[9,10,20,21] For each state, the sum of squares of the first few (or last few) principal components is computed. The first five principal components are used in this example. The 51 sum of squares are ordered and plotted against values of a gamma variable with suitably chosen shape and scale parameters. As shown in Figure 12, Alaska and New York--and especially California and Hawaii--appear to have come from a different population. It should be noted that these sum of squares are correlated regardless of the correlation structure of the original data. Hence, interpretation of results should be made very carefully.

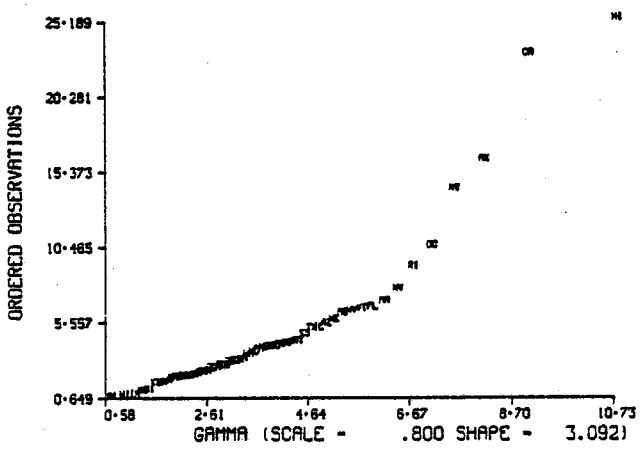


Figure 12

Figures 13, 14 and 15 show the so-called Chernoff faces for the 50 states plus the District of Columbia.[5] Here, a facial characteristic is associated with a variable as indicated in Table II. For example, wide noses correspond to large single populations and long noses correspond to large populations. The faces for New York and California are striking because of this feature. Similarly, Alaska has a wide face because of the large HHBTU consumption per capita, whereas Hawaii has a thin face.

TABLE II

Facial Characteristic	Variable
1. Face Width	HHBTU
2. Brow Length	SINGLE
3. Face Height	MAXT
4. Eye Separation	LAT
5. Pupil Position	AVEIN
6. Nose Length	POP
7. Nose Width	ONEP
8. Ear Diameter	PCURB
9. Ear Level	COML
10. Mouth Length	DEGD
11. Eye Slant	MEDIN
12. Mouth Curvature	PCAIR
13. Mouth Level	FRZR
14. Eye Level	LOWIN
15. Brow Height	OLDHS
16. Eye Eccentricity	LONG
17. Eyebrow Angle	NEWHS

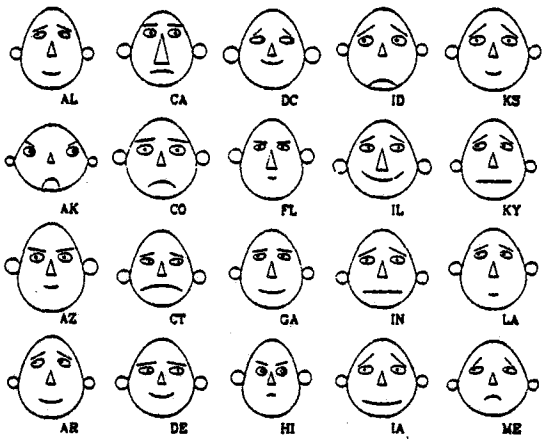


Figure 13

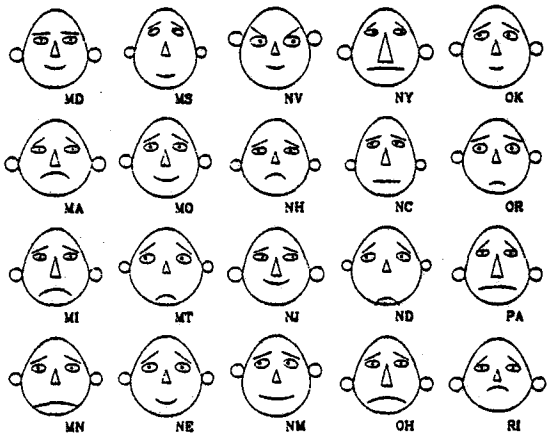


Figure 14

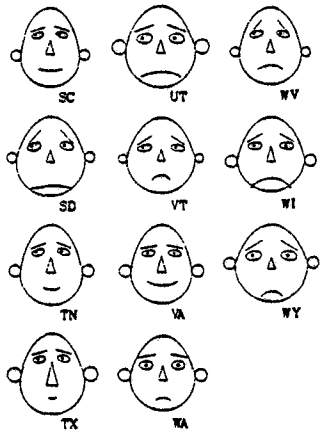


Figure 15

The dendrogram, a tree-like graph of non-overlapping hierarchical partitions, is another visual technique used in cluster analysis. A computer program containing eight clustering techniques (nearest neighbor, furthest neighbor, simple average, group average, median, centroid, Lance and Williams' flexible strategy, and Ward's method) is used. Initially, the data are standardized; both classical and robust standardization techniques are used. Regardless of the standardization and algorithm used, Alaska, California, Hawaii, and New York are distinct from the main cluster.

III. LARGE DATA SETS

In data analysis many of the problems can be attributed to the data itself, such as perhaps inaccurate, missing, too little, and recently too much. These large data sets not only create a tremendous storage problem, but challenge computer graphics for effective display techniques.

The analyses considered here deal with National Uranium Resource Evaluation (NURE) data. The objective of this nationwide airborne and stream sediment reconnaissance survey is to classify regions with respect to their potential mineralization. For example, in the stream sediment survey, LASL analyzes the data from five states: Wyoming, Colorado, Montana, New Mexico and Alaska. In the second year of a five-year study, LASL data bases already contain seven million words. Graphical techniques presented here include scattergrams, 3-D and 2-D density plots, a linear discriminant analysis display, contour maps, and moving windows.

The probability distributions of certain random variables such as thallium signals over a given geological formation of a flight line are thought to indicate uranium concentration. Figure 16 is a scattergram of bismuth vs. thallium for all geological formations on one map line in the Lubbock-Plainview area in Texas. The data in the lower left-hand corner represent recent geological formations and most of the formations follow a linear trend except for the data on the right-hand side of the plot where TL becomes constant with BI increasing. These data belong to two older formations with known uranium mineralization. Figure 17 shows data for the QT geologic formation.

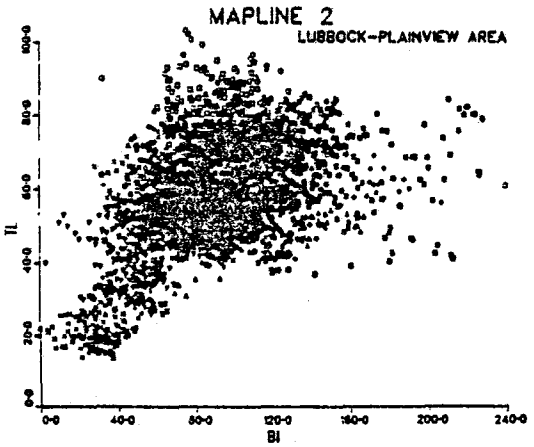


Figure 16

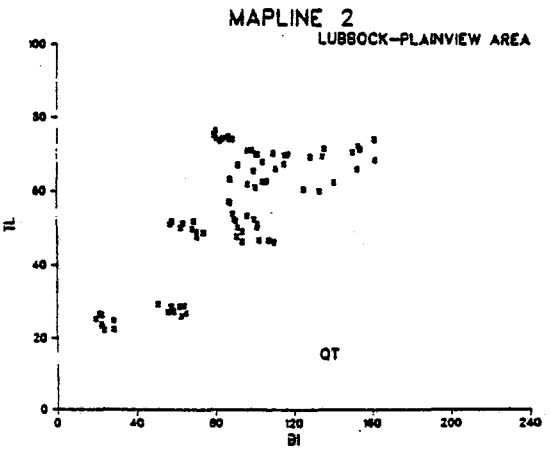


Figure 17

Scattergrams such as this are useful in identifying clusters representing misclassified geological formations data.

A technique for computing an empirical density function (edf) used to estimate a probability density function has been developed. As many as 100 of these densities, each representing a map line or transect, can be displayed simultaneously as shown in Figure 18. Since some of the edfs may be visually obscured by other edfs, the 3-D plots have been compressed into a 2-D grid plane in a lightness-darkness plot shown in Figure 19.

Figure 20 shows a linear discriminant analysis displayed as a gray-level matrix useful in delineating favorable and unfavorable regions of uranium mineralization. Each square represents 100 records (i.e., 100 seconds of gamma-ray signals on a map line) in the Lubbock area. The 23 rows represent 23 map lines. There are eight gray levels which are linearly spaced from light to dark over the interval [0,1]. The lighter shades represent low probability of favorable uranium mineralization while darker shades represent high probability of favorable uranium mineralization.

Contour maps of the radiometric variables are also useful in indicating regions of favorable mineralization. A contour map of the bismuth-thallium ratio in the Lubbock-Plainview area is shown in Figure 21.

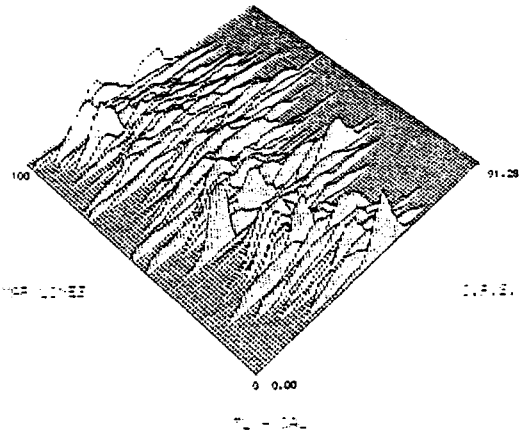


Figure 18

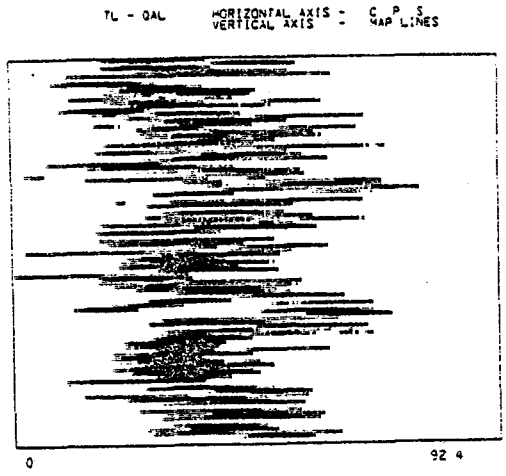


Figure 19

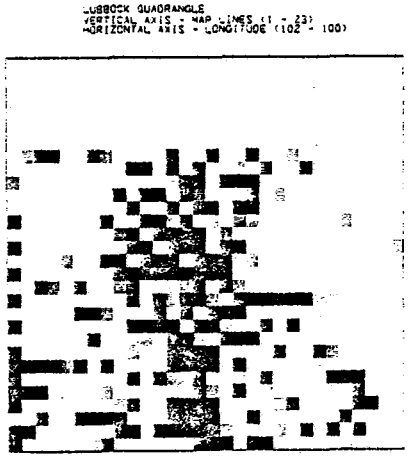
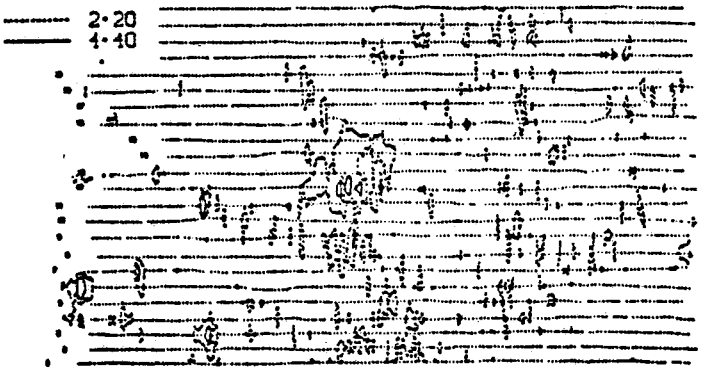


Figure 20



Bismuth-Thallium ratio.

Figure 21

Figure 22 shows a method for detecting clusters of bismuth anomalies. The map of the anomalies in the Lubbock area represents a 2-D Poisson process. A rectangular moving window 6 miles long and 8.5 miles (or 3 map lines) wide is used to identify clusters containing 5, 6, and 7 or more anomalies at a specified probability level.

Figures 16-22 represent different ways of displaying data analysis techniques. It is interesting to check whether these displays indicate the same favorable areas of potential mineralization.

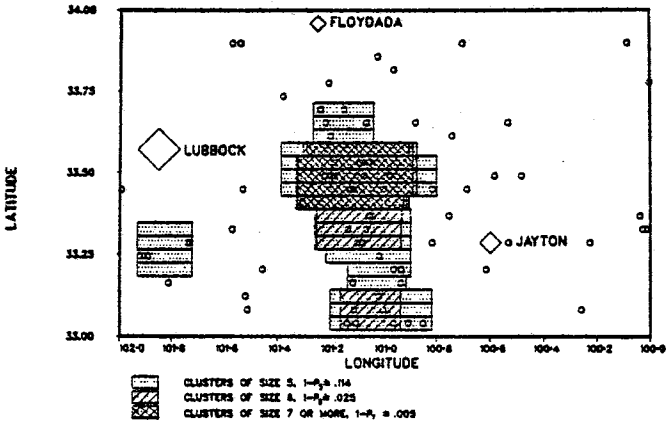


Figure 22

IV. CARTOGRAPHIC DATA SETS

Maps are very useful in displaying and communicating information contained in data with spatial/geographic relationships. The figures shown are applications of cartographic techniques and have been extracted from various on-going projects.

Figure 23 summarizes U.S. offshore oil and gas lease data from October 1954 - November 1976. The number of leases, the leasing years, the acreage and the producing acres through 1974 are given for individual states and regions as well as totals for all the leases. Of the total 2,260,000 producing acres, Louisiana has 2,116,000 acres and Texas has 118,000 acres.

Figures 24 - 26 are for a study of the impacts of electric power generation in the West. The location of existing and proposed power plants by type for the Western and Rocky Mountain regions are shown in Figure 24. The letters represent the type of plant, i.e., coal, oil, gas and nuclear. The size of the letters indicate three levels of power generation: small, 500-999 MWe, medium, 1000-1999 MWe, and large, 2000+ MWe. The Los Angeles and San Francisco areas have a number of oil-fired plants and these areas are simply shaded. Figs. 25 and 26 are maps used to study pollution dispersion patterns. Figure 25 shows SO_2 concentration in southwest Wyoming for 1985 with pollution contours drawn every 0.25 $\mu g/m^3$. Figure 26 shows change in length of life due to pollution in days per person. Similar graphical displays were made for exposure to suspended particulates, additional restricted activity days due to pollution and annual morbidity costs per person and per town.

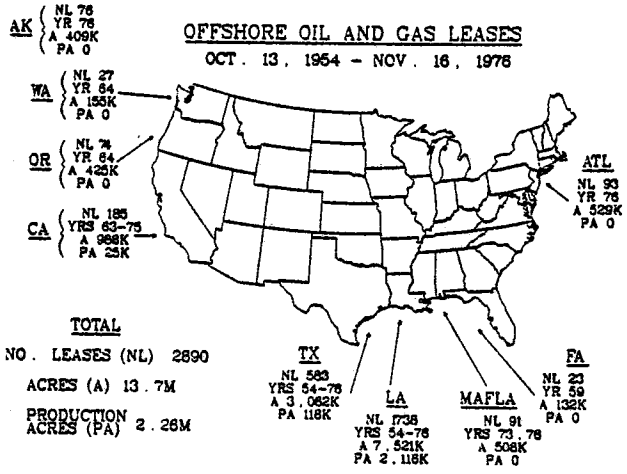


Figure 23

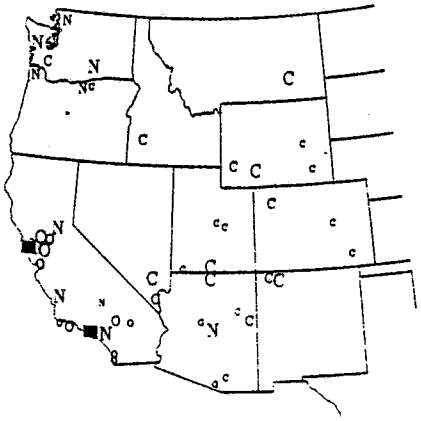


Figure 24

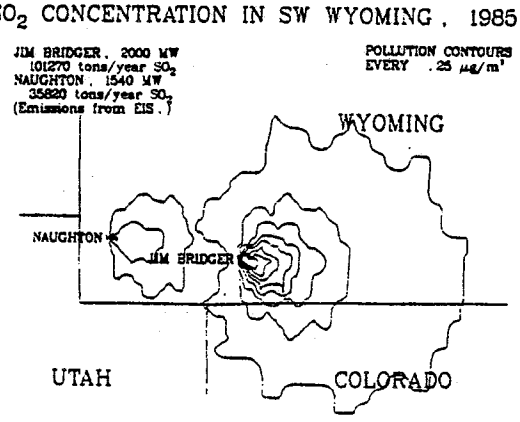


Figure 25

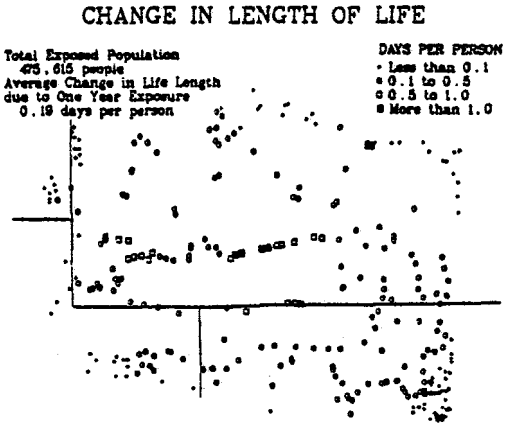


Figure 26

Computer-generated images aid in the study of the effect of various contaminants on visibility. Photographs of landscapes have been digitized on a microdensitometer in wavelengths corresponding to red, blue and green light. Data on transmitted and total radiation are calculated and converted to equivalent densities using a plume visibility code. The densities are superimposed on the image to form the pollution cloud. The color and extent of the cloud are determined by the type of pollutant, different control technologies and varying meteorological conditions.

Figures 27 - 29 are from solar feasibility studies.[16] The first map shows heating degree days which is the average of the high and low temperatures subtracted from a 65° base temperature summed over 365 days for one city in each of the 48 states. Simply stated, the colder the climate, the higher the number of heating degree days. Contrast Florida with 214 and Maine with 7511. The second map shows 1977 residential gas prices in dollars per thousand cubic feet by state. Gas is generally cheaper in the southern, central and Rocky Mountain regions. Note that Maine has higher prices than nearby Vermont and New Hampshire. Figure 29 shows the pattern of economic feasibility for domestic hot water without incentives, with incentives provided by the National Energy Plan of April 1977, and the House Modification of that plan.

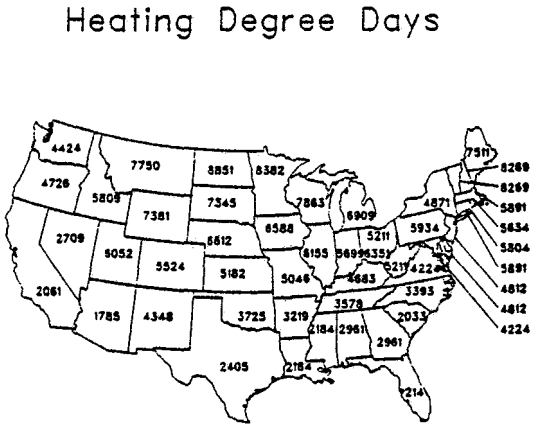


Figure 27

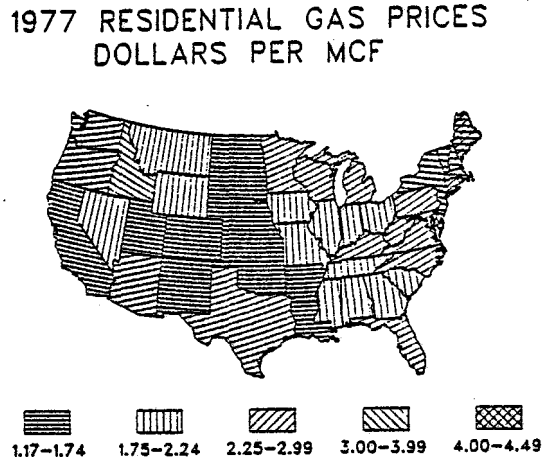


Figure 28

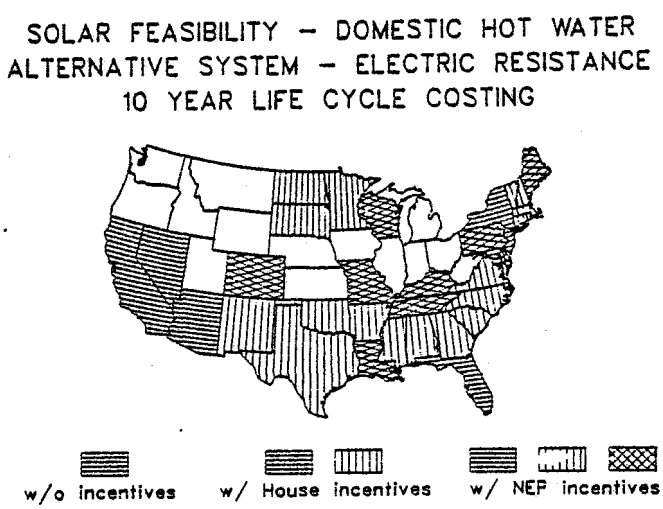


Figure 29

Figures 30 - 37 are maps displaying energy-related data from the Regional Studies Program. Figure 30 shows five coal export-import regions and Figure 31 is a flow map for the export of Rocky Mountain coal. The circle represents the within region total and the thickness of the arrows represents relative amounts of export to the other four regions. Bar charts and pie charts are useful in displaying energy totals for regions or states. Production, consumption, export and negative export (import) figures are displayed in Figures 32 and 33 using shaded bars. Figure 34 uses varying sized circles to indicate production levels by region. Sections of a circle are shaded differently to indicate coal, oil and natural gas, hydro, nuclear and other and uranium production. Figure 35 shows county air quality maintenance area data for the Rocky Mountain region.

An interactive composite geo-information mapping system known as GMAPS provides map data on such items as wilderness areas, ecosystem trends, locations of natural resources, etc., for selected regions in the U.S. Figure 36 shows types of coal fields and Figure 37 is a composite of coal fields with oil shale basins in the Rocky Mountain region.

COAL EXPORT–IMPORT REGIONS

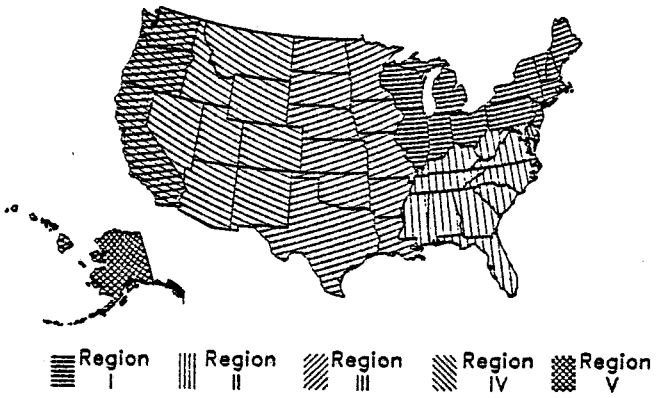


Figure 30

ROCKY MOUNTAIN COAL

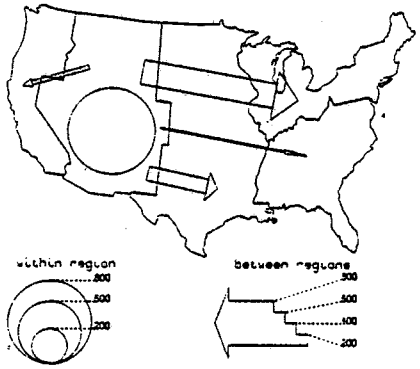


Figure 31

1975 REGIONAL ENERGY TOTALS

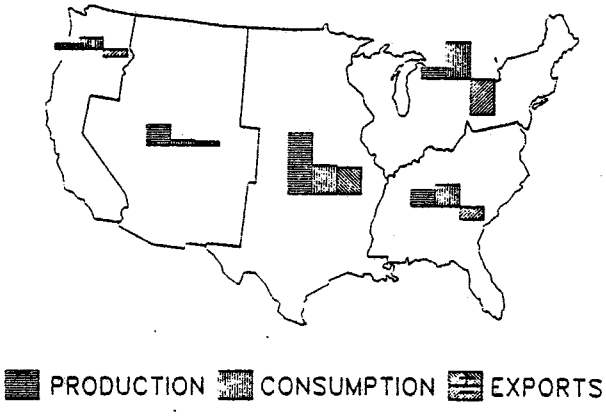


Figure 32

ROCKY MT. COAL

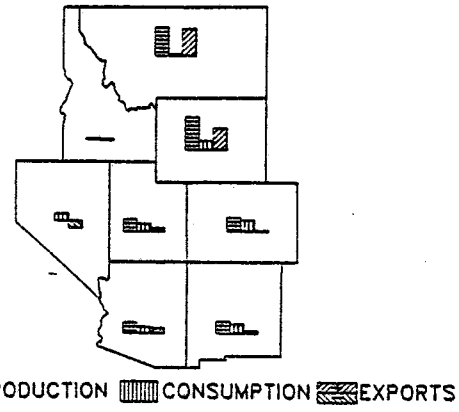


Figure 33

1975 REGIONAL ENERGY PRODUCTION

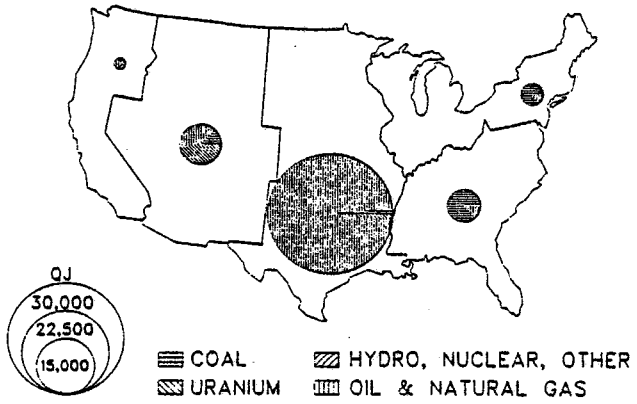


Figure 34

AIR QUALITY MAINTENANCE AREAS
ROCKY MOUNTAIN STATES

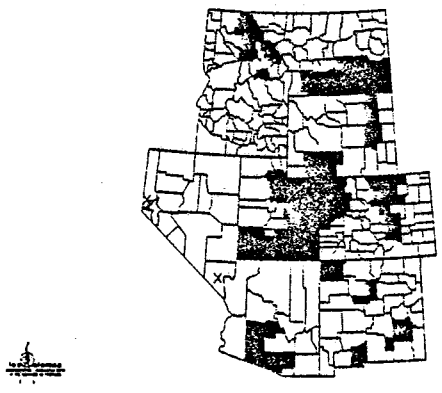


Figure 35

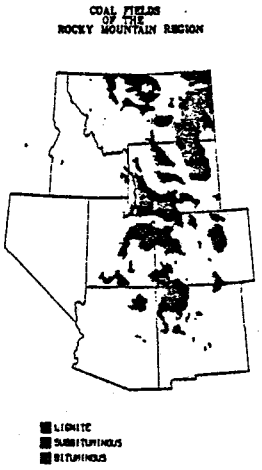


Figure 36

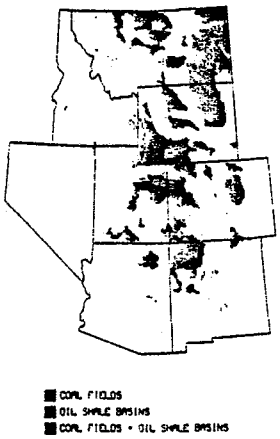


Figure 37

V. SUMMARY

The computer-generated graphic products described in this paper represent a variety of techniques for displaying and analyzing small univariate and multivariate data sets, large data sets and cartographic data sets. Computer graphics are useful tools for communicating information efficiently and effectively.

ACKNOWLEDGMENTS

We wish to thank Katherine Campbell and Mona Wecksung for the use of some of their slides, and Thomas Bement and John Sibert for assistance in coding. Melvin Prueitt supplied the 3-D plotting program, PICTURE.

BIBLIOGRAPHY

1. D.F. Andrews, "Plots of High Dimensional Data," Biometrics, 28, 125-136 (March 1972).

2. T.R. Bement, D.V. Susco, D.E. Whiteman, R.K. Zeigler, "National Uranium Resource Evaluation Program," Los Alamos Scientific Laboratory Report LA-6804-PR (May 1977).

3. L.A Bruckner, M.M. Johnson, G.L. Tietjen, "The Analysis of Lease, Production and Revenue Data from Offshore Oil and Gas Leases," Los Alamos Scientific Laboratory paper presented at the Second ERDA Statistical Symposium, Oak Ridge, TN (October 1976).

4. K. Campbell, "IMGPROC - An Image Processing Program for CDC 6600 and 7600 Computers," Los Alamos, NM (November 1974).

5. H. Chernoff, "The Use of Faces to Represent Points in K-Dimensional Space Graphically," Journal of the American Statistical Association, 68, 361-368 (June 1973).

6. W.J. Conover, Practical Nonparametric Statistics (John Wiley & Sons, Inc. New York, 1974).

7. W.J. Conover, T.R. Bement, R.L. Imen, "On a Method for Detecting Clusters of Possible Uranium Deposits," submitted to Technometrics.

8. A. Ford and H.W. Lorber, "Methodology for the Analysis of the Impacts of Electric Power Production in the West," Los Alamos Scientific Laboratory Report LA-6720-PR (January 1977).

9. R. Gnanadesikan and J.R. Kettenring, "Robust Estimates, Residuals, and Outlier Detection with Multiresponse Data," Biometrics, 28, 81-124 (March 1972).

10. R. Gnanadesikan, Methods for Statistical Data Analysis of Multivariate Observations (John Wiley & Sons, Inc., New York, 1977).

11. G.N. Lance and W.T. Williams, "A General Theory of Classificatory Sorting Strategies. I. Hierarchical Systems," Computer J., 9, 373-380 (1967).

12. E.M. Leonard, M.D. Williams, J.P. Mutschlecner, "The Visibility Issue in the Rocky Mountain West," Los Alamos Scientific Laboratory Report LA-UR-77-2558 (September 1977).

13. R.K. Lohrding, "Three Kolmogorov-Smirnov-Type One-Sample Tests with Improved Power Properties," J. Statist. Comput. Simul., 2, 139-148 (1973).

14. R.K. Lohrding, "Statistical Analysis and Display of Energy-Related Data," Los Alamos Scientific Laboratory paper presented at the ERDA-Wide Conference on Computer Support of Environmental Science and Analysis, Albuquerque, NM (July 1975).

15. R.K. Lohrding, "Comparative Power Studies of Some Tests of Normality," Los Alamos Scientific Laboratory Report LA-5101-MS (November 1972).

16. F. Roach, "Prospects for Solar Energy: The Impact of the National Energy Plan," Los Alamos Scientific Laboratory Report LA-7064-MS.

17. R.A. Waller, E.A. Monash, and J. Lohrenz, "Some Computerized Graphic Technical Applications for Federal Mineral Lease Management Support," Los Alamos Scientific Laboratory paper presented at the First Computer Symposium, Reston, VA (March 1977).

18. J.H. Ward, Jr., "Hierarchical Grouping to Optimize an Objective Function," J. Amer. Statist. Assoc., 58, No. 301, 236-244 (1963).

19. M. Wecksung, R. Wiley, and K. Turner, "GMAPS User's Manual," Los Alamos Scientific Laboratory Report LA-6975-M.

20. M.B. Wilk, R. Gnanadesikan, M.J. Huyett, "Estimation of Parameters of the Gamma Distribution Using Order Statistics," Biometrika, 49, Nos. 3 and 4, 525-545 (1962).

21. M.B. Wilk, R. Gnanadesikan, M.J. Huyett, "Probability Plots for the Gamma Distribution," Technometrics, 4, No. 1, 1-20 (February 1962).

22. J.W. Wood, "A Computer Program for Hierarchical Cluster Analysis," Newsletter of Computer Archaeology, 9, No. 4, 1-11 (June 1974).

AN INTERACTIVE GRAPHICAL DATA ANALYSIS SYSTEM

Richard L. Phillips
The University of Michigan, Ann Arbor, Michigan, 48109

ABSTRACT

ADROIT (Automated Data Retrieval and Operations Involving Timeseries) is an interactive system which is capable of rapid retrieval, statistical processing and graphical display of large data bases. ADROIT comprises two major subsystems, the computational subsystem (ACS) and the display subsystem (ADS). The heart of ACS is an interpretive programming language which has been designed to handle time series data types. Besides the usual data types found in programming languages like FORTRAN, ADROIT includes an observation and a time interval type. These permit proper statistical operations to be performed and aggregation of data to arbitrary time periods and intervals. ADS allows the user to design and produce report-ready graphs of data processed by ACS. Among the display options available to the user are: point, line, or bar graph plotting of data; curve smoothing or least squares data fitting; absolute, relative or cumulative histograms.

INTRODUCTION

There is no argument about the value of a graphical display as an aid in discovering relationships hidden in complex datasets. Such a discovery process is, however, inherently iterative, meaning that interactive graphics is required. A system has been developed that permits an analyst to interactively manipulate and analyze arbitrary datasets and to display the results on dynamically designed graph backgrounds. What follows is a detailed discussion of this system, explained in the context of a particular dataset — one dealing with water quality. It should be understood, however, that the system is general and can and has been used with many other types of datasets.

THE COLLECTION OF WATER QUALITY DATA

On a more or less periodic basis, state governmental agencies responsible for enforcement of water quality standards in the United States take samples of the inland waters under their jurisdiction. These samples are taken at carefully chosen locations called stations; each station has its own unique identifying code called a station number.

The samples are chemically analyzed to yield potentially over 2500 different so-called parameter values, numerical values describing the goodness of the water; examples of these parameters are dissolved oxygen level, nitrate concentration, and mercury concentration. Each parameter has its own unique identifying code called a parameter number.

Each parameter value has a "time of observation" associated with it, given in a form known as the EPA date. The EPA date is a ten digit number composed of five, 2-digit fields: the last two digits of the year, the month, the day, the hour, and the minute of the data sample. For example, 1:22 pm on December 30, 1976 is denoted as 7612301322.

THE STORET SYSTEM

Combined with physical parameter values such as stream flow and water temperature, the chemical parameter values from a sampling expedition are sent to the United States Environmental Protection Agency (EPA) to be stored in a large computer data bank called STORET [1]. The data in the STORET system are (parameter value, EPA date) pairs,

catalogued by station number and parameter number. In the State of Michigan alone, over 150 parameter values are collected at 718 stations. In spite of the fact that inclement weather frequently prevents sampling at monthly intervals, STORET now contains about 3-million useful (parameter value, EPA date) pairs for Michigan. When this number is multiplied by the number of states participating in the program, the amount of data involved becomes truly prodigious.

Leaving aside the question of the physical management of the data bank, the problem remains that there is no sense in collecting all the data in STORET unless someone analyzes it. STORET can currently produce preprogrammed (i.e. "canned") high-level, statistical summary reports in batch mode, but it has no time series analysis capabilities, and it has no interactive information retrieval capabilities nor any interactive computational capabilities. The logistics problems of retrieving and working with STORET data impede exploratory, qualitative studies which seek to go beyond the preprogrammed summary reports, and make rigorous, quantitative studies very difficult.

The development of the ADROIT system [2] (Automated Data Retrieval and Operations Involving Timeseries) was undertaken to alleviate the shortcomings of the STORET system for use within the State of Michigan. It was clear that a water quality investigator could realize significant gains in his capabilities and his productivity if he could interactively pose queries to the data bank using familiar notation, interactively perform statistical analyses of the retrieved data, and graph the results of such analyses on an interactive-computer-graphics terminal. In addition, it was recognized that ad-hoc procedures for handling the large quantities of data involved would likely be doomed to failure. In short, the highest returns seemed to be in an interactive, systematic problem-solving environment specifically tailored to retrospective analysis of the water quality time-series data in STORET. ADROIT attempts to provide such an environment.

THE ADROIT SYSTEM

ADROIT is a system of integrated computer programs which have been designed to perform rapid retrieval and statistical analysis of water quality time-series data, and to produce report-ready graphs of selected results. ADROIT caters both to the casual browser who is taking a cursory look at a small data set and to the more serious investigator rigorously studying a large data set. For casual and exploratory studies, ADROIT is usually run completely interactively. After an analysis scheme has been interactively developed and verified, further analyses and/or plotting of results may be performed without user intervention.

Particular attention was paid throughout the design of ADROIT to human engineering problems — e.g. selection of defaults, extensive and early diagnosis of errors, elimination of error-prone constructions, and production of readable error messages. The result is a system which because of extensive defaulting is easy to use for everyday tasks, but in which it is possible to specify operations in very fine detail if the user wishes.

ADROIT is currently implemented as two intercommunicating interpreters, each of which communicates with the user through its own special-purpose programming language. The interpreters are known as the ADROIT Computational Subsystem (ACS) and the ADROIT Display Subsystem (ADS).

THE ADROIT COMPUTATIONAL SUBSYSTEM

A great deal of design effort was expended in developing a rudimentary but acceptably complete calculus of time series for the ACS Language, to provide a focus for the analysis process. The ACS interpreter in turn implements this calculus. Thus the user need not change notations between the step of conceptualizing an analysis scheme and the step of instructing the computer to perform an analysis.

The ACS Language resembles more common, general-purpose programming languages, but error-prone features (such as the GO TO statement and implicit type conversions) are not present, and new data types and built-in functions which facilitate the analysis of time-series data have been added. The definitions of arithmetic operators have been extended to the new data types, and vector

arithmetic similar to that available in APL [3] has been implemented. Thus a few lines of ACS Language can quite often specify the computations expressed in several hundred lines of FORTRAN or ALGOL.

The concept of data typing plays a significant role in the ACS Language. Every constant, variable, and expression always has a well-defined type. The familiar data types NUMERIC, LOGICAL and STRING are present, as well as three new data types: the numeric interval NUMINT, the time interval TIMEINT, and the observed time series OBS. When requested to perform an operation involving one or more expressions, the interpreter checks the type of each expression to ensure conformance to the defining rules for that operation. Thus for example, the interpreter generates an error comment when the user tries to multiply a time interval by a time series (TIMEINT times OBS) because such an operation is not defined in the ACS Language. In such a case, other programming languages often try to do an automatic type conversion and resume computation, producing astonishing results in the process!

Since the users of ADROIT are familiar with EPA date notation, a constant of type TIMEINT is given in those terms; for example,

TIME 7507010C0 THRU 7512 BY 2 MO

is a constant of type TIMEINT and denotes the interval from the first minute of July 1, 1975 through the end of 1975, partitioned into 2-month intervals. ACS understands abbreviations in the specification of EPA dates; e.g. "7512" above. Variables may be declared to be of type TIMEINT, and as with other programming languages, such variables may be used anywhere in the language where an expression of type TIMEINT is required.

A datum of type OBS, i.e. a time series, consists of a 4-tuple of vectors: the sample mean, the sample variance, the sample weight, and the time of observation. A time series of (parameter value, EPA date) pairs as retrieved from the data bank is expanded by ADROIT into a constant of type OBS as follows. Suppose there are N pairs in the retrieved time series. Then the interpreter allocates four vectors, each N elements long. (The user need not prespecify vector length information as done in FORTRAN; storage allocation for all data types is done automatically.) Into the sample-mean vector are put all of the parameter values for the time series. The time-of-observation vector holds the EPA dates corresponding to the parameter values. Each element in the sample-variance vector is set to zero to indicate that the corresponding mean element is known "exactly" (i.e. to within laboratory error, which is ignored). Each element in the sample-weight vector is set to one to indicate that the information from only one parameter value is reflected in the corresponding sample-mean and sample-variance elements.

Before time-series data can be retrieved from the data bank, a particular station is selected by assigning its station number to the reserved STRING variable CURSTA. Time series at this particular station may then be retrieved by a date-retrieval variable, denoted by prefixing a parameter number by the letter "P". For example, P300 denotes dissolved oxygen since the parameter number for dissolved oxygen is 300. A data-retrieval variable has the data type OBS and may be used anywhere in the language where an expression of type OBS is required.

The ADROIT user manipulates time series through built-in functions which operate on time series to produce new time series. Typical of such a function is AGG. which is used to pool data to achieve some measure of statistical reliability. A typical invocation of AGG. is

AGG.(P300, TIME 64 THRU 74 BY 1 YR)

The operation of this AGG. call is as follows. First, eleven aggregation intervals as specified by the second argument are computed; these would be the year of 1964, the year of 1965, and so on. Then for each aggregation interval, one element of the resultant OBS value of the AGG. function is computed as follows. The mean element is the average of all mean elements in P300 which fall into the aggregation interval, weighted by their corresponding sample weights. The variance element is a combination of all the variance elements in P300 which fall into the aggregation interval, weighted by their corresponding

sample weights (the proper statistical formula is used). The sample-weight element is the sum of all the sample-weight elements which fall into the aggregation interval. The time element is the average of the beginning and the end of the time interval. Thus the result of this AGG. call is a time series of eleven points and the mean element of each point is a level of dissolved oxygen, averaged over a one-year period. Note that the result can be further aggregated or operated upon by any other function which operates on data of type OBS.

Other time-series manipulation functions in the ACS Language handle problems of simultaneity; although water quality parameters change slowly, it is hardly supportable to multiply an effluent concentration sampled in the winter by a flow sampled in the summer to obtain an effluent mass value. Thus all the time series which are to participate in a computation are first moved onto a common time net. The common time net of a set of time series is an ordered set of time values such that for any time value in the net, there is a corresponding time-series value in each of the time series. The ACS function COMMTIME. operates on a variable number of time-vector constituents of time series to produce a NUMERIC VECTOR datum which is the common time net of the time series. This common time net can then be used by the EXTR. function to extract the common points from time series for further computations. In actuality, the requirement for strict simultaneity is relaxed somewhat; if two time points lie within a user-specified number of minutes of each other, they are adjudged to be simultaneous as far as COMMTIME. is concerned.

In addition to the time-series manipulation functions, there are type-conversion functions, standard numerical functions like square root and logarithm, a function to sum the elements of a vector, hydrological utility functions, and statistical functions which compute the inverse normal, chi-squared, Fisher's F, and Student's t distributions. These functions provide building blocks for arbitrarily complex analysis procedures which can be developed by the user.

ACS has the capability of utilizing a library of user-defined procedures. Thus when the user finds there are sequences of statements he frequently performs and finds useful, he can catalog them as a procedure by giving them a unique name. When the procedure is to be invoked, the user simply types its name (and any appropriate arguments), and ACS executes the procedure immediately.

A complete set of statements to direct the flow of control adds to the power of user-defined procedures. In addition to the IF statement, ACS Language provides FOR and WHILE statements similar to those of ALGOL to handle loops. An ABORT statement causes an error return and procedure traceback to appear on the user's console. Control can be transferred from the ACS to the ADS (display subsystem) by a GRAPH statement, followed optionally by a list of arguments.

A pair of examples will perhaps clarify the above discussion. Suppose the task at hand is to plot the soluble ortho-phosphate concentration at the mouth of the Grand River, aggregated by year, from 1963 through 1974, with 80% confidence intervals shown on the plot. The following lines, typed at the user's interactive graphics terminal, will produce the desired plot.

```
CURSTA = '700026'
TIMPER = TIME 63 THRU 74 BY 1 YR
GRAPH AGG.(P70507,TIMPER)
AUTO
DATA
LINE
CONF 0.8
```

The first line specifies the station number of the station at which time series will be retrieved. The second line assigns the time period of interest to the variable TIMPER, which is gratuitously predeclared in the interpreter to be of type TIMEINT for just this use. The third line causes the ADS to be invoked to graph the time series which is the result of the aggregation of the soluble ortho-phosphate concentration P70507 over the time period. The remaining lines are instructions to the ADS which cause AUTOmatic graph scaling for a plot of (the aggregated) parameter value versus time, plotting of the DATA in the sample mean vector, connection

of adjacent data points by a straight LINE, and plotting of the CONFidence limits (assuming Gaussian distributions and based on the sample variance and Student's t statistic). The graphical output is shown in Figure 1.

As a second example, consider the following problem: plot with 80% confidence limits the soluble ortho-phosphate mass (called a "loading") in pounds per day flowing by the same station over the same time period, aggregated by year. The solution will be cast in terms of a user-defined procedure named PHOSLOAD. which computes the soluble ortho-phosphate loading at station CURSTA during time period TIMPER. To invoke PHOSLOAD. to solve the problem (assuming control is still situated within the ADS from the first example), the user types:

```
EXIT
GRAPH AGG.(PHOSLOAD.,TIMPER)
AUTO
DATA
LINE
CONF 0.8
```

The first line causes the ADS to return control back to the ACS. The second line reinvokes the ADS to plot the value of the procedure PHOSLOAD. aggregated over the time period. Note that the variables CURSTA and TIMPER retain their old values and need not be reassigned. The remaining lines are the same as for the previous example. The graphical output is shown in Figure 2.

The text of the user-defined procedure PHOSLOAD. is shown in Figure 3. Both reserved words and built-in functions in the ACS Language are shown in lower case, and reserved words are underlined. Variable names are given in upper case. The percent character "%" is used to start comment lines, while a trailing dash "-" on a line implies that the statement is continued on the next line. In statement 1, the procedure name is declared; since no arguments are specified after the procedure name, the interpreter will check to see that the user supplied no arguments in the invocation. Statements 2 and 3 declare the temporary variables which will be used in the computation. Three variables of type OBS are declared, and one VECTOR variable of NUMERIC type is declared to hold the common time net for t computation. (All variables must be dec their first use to allow the inte perform complete type checking.) ments 4 and 5, the time series fo ortho-phosphate concentration (P7 mg./liter and stream flow (P60) i are retrieved; the RESTRICT. func cards all points lying outside th od of interest; and the (possibly resulting time series are stored in variables P and F. In stateme COMMTIME. is used to compute the net; the TIME. function is a type function which operates on a time extract the time-of-observation v return it as a datum of type NUME In statement 7, the actual loadin puted. The constants which appea sion factors to render the units into pounds per day. Statement 8 interpreter to return control to of PHOSLOAD., returning the curre the variable LOAD as the value of dure. Note that the value of PHO thus a time series. Statement 9 serves as a marker for the end of cedure.

red before reter to n state- soluble 07) in cu ft/sec on dis- time peri- maller) mporarily . 6, mmon time onversion eries to tor and C VECTOR. is com- are conver- the answer irects the e caller value of he proce- OAD. is mply he pro-

THE ADROIT DISPLAY SUBSYSTEM

Just as a calculus of time se lies the manipulation of STORET d ACS, a rigorous, canonical defini graph is a fundamental aspect of The canonical graph description i by the ADS in a form called the g ture. The graph structure is an of graph elements, each of which by a number of parameters. The e into either the background group with scaling and labeling or the which is generated from the data plotted. The background group of ments includes the x-y axis syste and y-axis tick marks, the x- and lines, labels for the x- and y-ax marks, and the titles for the x a The data group includes the data plotted on the graph by plotting the solid and dashed lines plotte graph, the smoothed curve plotted graph, the regression line plotte graph, the bar graph plotted on t

es under- a in the on of a e ADS. maintained ph struc- dered list described ments fall sociated ta group ints to be raph ele- the x- -axis grid tick y axes. ints aracters, on the n the on the graph,

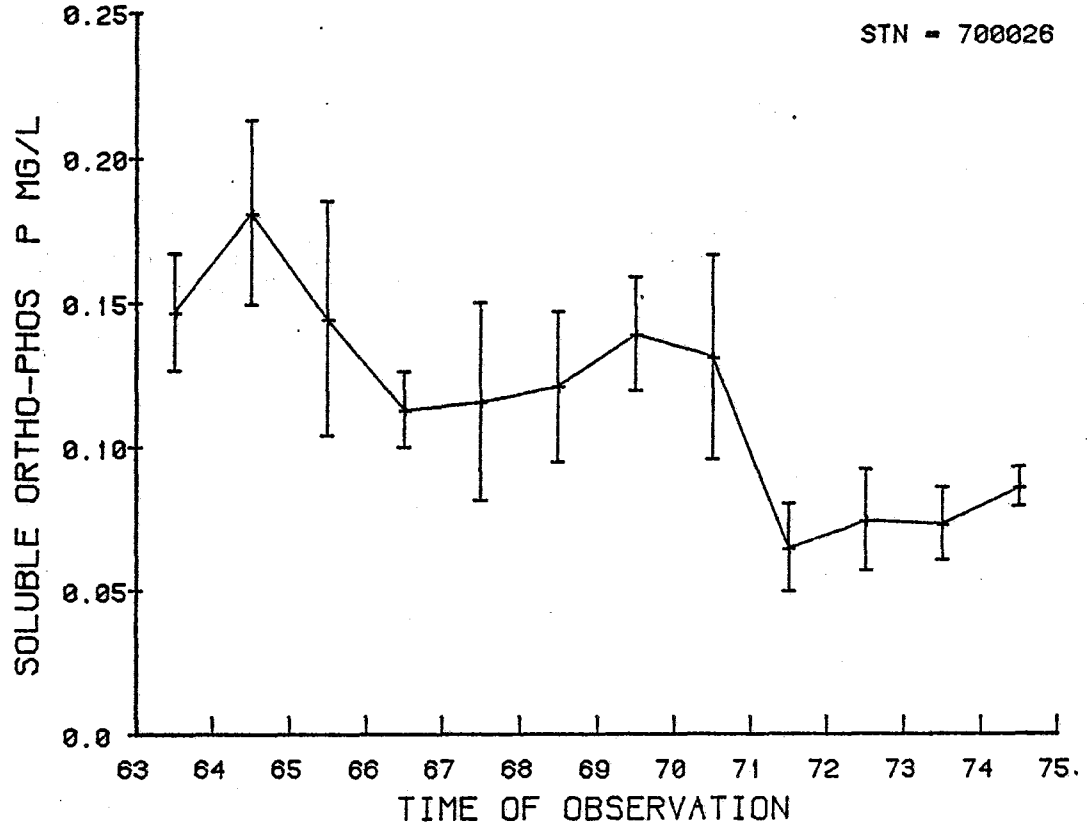


Figure 1. Plot output for example 1.

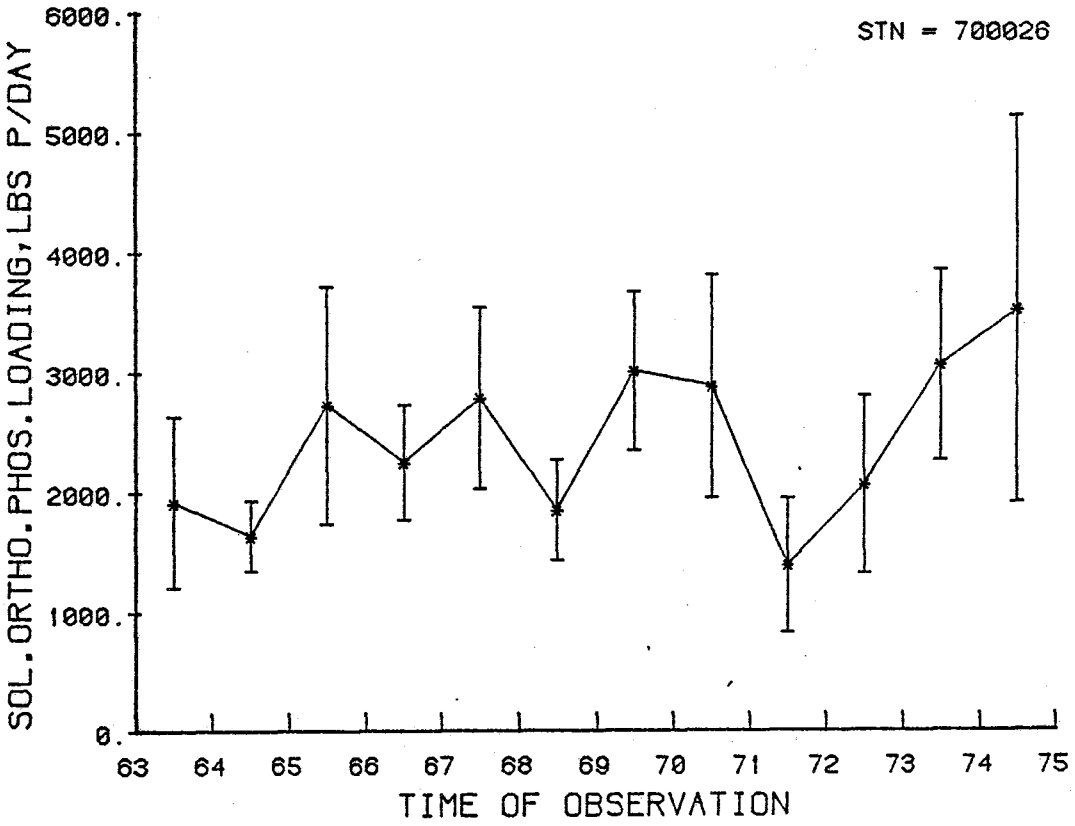


Figure 2. Plot output for example 2.

Statement	Text of Procedure
1	procedure PHOSLOAD.
	% The declarations for local variables follow
2	obs P, F, LOAD
3	numeric vector NET
	% The executable part of the procedure follows
4	P = restrict.(P70507,TIMPER)
5	F = restrict.(P60,TIMPER)
6	NET = commtime.(time.(P),time(F))
7	LOAD = 2.2045E-6 * 28.316 * 86400 * -
	extr.(P,NET) * extr.(F,NET)
8	return LOAD
9	endproc

Figure 3. The user-defined procedure PHOSLOAD

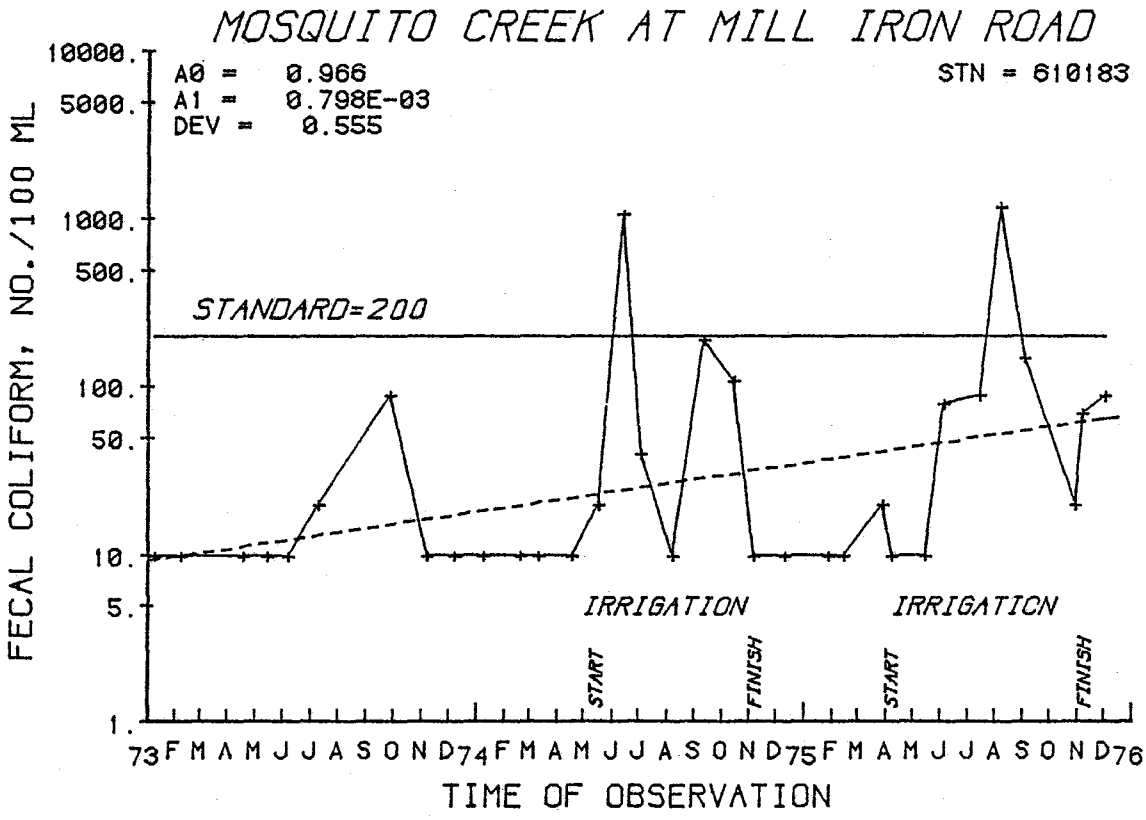


Figure 4. Example of graph background

and general textual annotation applied to the graph. Each of the graph elements is under control of the user, and may in fact be omitted.

The user controls the graph structure through a simple set of keyword commands. For example, the user can specify the placement of the tick marks on the x axis by

TICX = 0.0, 2.0

which indicates that tick marks are to begin at x = 0.0 and continue thereafter with a spacing of 2.0, i.e. x-axis tick marks are to appear at 0, 2, 4, 6, . . .

If the user wishes, he can avoid specifying graph element parameters explicitly by using the AUTO command. If this command is issued, the ADS will use the extrema of the data currently being plotted to compute scaling factors to generate a graph structure which will produce a visually pleasing graph. The AUTO command produces axes, tick marks, tick mark labels, and if appropriate, axis titles. In most cases, the graph produced by the AUTO command will be acceptable, but if not, the user can easily modify it. It is possible to use the ADS in a semi-automatic mode wherein the user specifies only those graph elements over which he wants control and then issues the AUTO command to let the ADS determine the rest of the elements.

Once the background group has been established, the user can plot his data in a variety of ways. He can select a plotting character for discrete data representation, or he can join the data by straight or dashed lines, or both. A smooth curve can be passed through the data, as if done by a draftsman, or a least-squares regression polynomial can be passed through imprecisely known data.

Data values can also be represented by a bar graph, with bar width and texture being under user control. Histograms of the current data can also be produced.

If the elements of the background group are frequently used, the user can name them and store them in a special file, to be called up later to plot the same or different data. Figure 4 is an example of data plotted on a frequently used graph background.

When the user has finally obtained a satisfactory graph of his data, he can either produce a Calcomp plot directly or store the grph in a so-called "holding file" for later post-processing by the COMPOSE subsystem.

THE COMPOSE SUBSYSTEM

The ADROIT system includes a graphical post-processing module called COMPOSE which is run as a stand-alone program. COMPOSE was designed to produce report-ready graphs by selectively combining and annotating individual graphs produced by the ADS.

COMPOSE is page-oriented. The user imagines he is preparing a figure for a report and specifies page size, orientation, and margin width. A composite graph may then be constructed using from one to six of the elementary graphs in the holding file. Elementary graphs may be selected from the holding file in any order and placed at any available position on the composite page.

After producing the desired combination and arrangement of graphs, the user can add more text and graphical enhancements to the page. The user may wish to add figure titles (as has been done with Figures 1 and 2), explanatory text in the body of the graphs, or legends that tie the graphs together. COMPOSE allows full control over character size, orientation, type font, and italics angle.

Graphical annotation takes the form of special symbols such as arrowheads, rectangles, regular polygons, and straight lines and arcs. Using the cross-hair cursor on the graphics terminal, the user can position these graphical enhancements anywhere on the page. To produce annotation and enhancement of a desired size, the user can zoom in on any part of the page, add graphics or text, and zoom out again to full page size.

Finally, a PLOT command can be issued to produce a Calcomp plot of the finished page.

SUMMARY

ADROIT has been in use in Michigan for over two years. During that time, a few bottlenecks in the design have been found, but the number (less than half a dozen) has been gratifyingly small. The system has been amazingly productive, exceeding everyone's

expectations. It is easy to learn (about a day for people familiar with STORET), and because it is extremely interactive, it is easy to use. The elapsed time and computer time cost to produce Figures 1 and 2, for example, was about fifteen minutes and $4.00. This compares favorably to the hours of hand computation and drafting that would be necessary using previous methods.

The author strongly feels that the success of ADROIT can be traced both to its emphasis on human factors and to the design effort expended to develop a systematic, coherent scheme for analyzing time series and producing graphs.

REFERENCES

1. Green, R.S., The Stroage and Retrieval of Data for Water Quality Control, U.S. Government Printing Office, Washington, D.C., August 1966.
2. "ADROIT - A System for Water Quality Data Analysis and Display," UNIDATA, Inc., Ann Arbor, Michigan, June 1975.
3. Iverson, K.E., A Programming Language, John Wiley and Sons, New York, 1962.

Data Structures

for

Cartographic Analysis and Display

W. R. Tobler

Department of Geography
University of California, Santa Barbara 93106

ABSTRACT

A variety of methods for the coding of geographical points, lines, areas, and surfaces are available. These permit the automatic production of geographic maps, but also permit direct spatial analysis without the use of visual illustrations (maps). The efficiency of the various met
has higher priority.

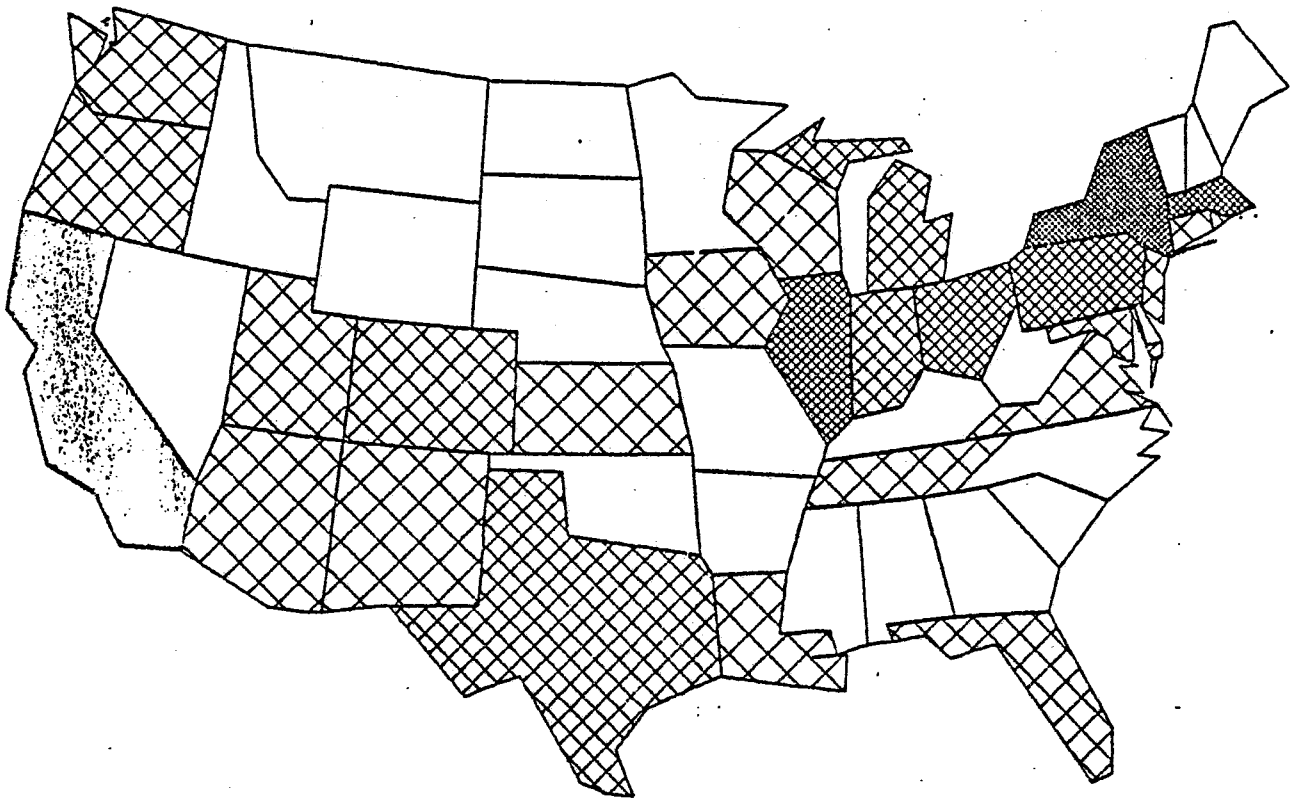
In this paper I attempt to provide an overview of the subject suggested by the title. A detailed account of particular data structures is not attempted; references are provided for this purpose. Nor is any particular data structure advocated; it is abundantly clear that the appropriate data structure depends on the application. To the extent that one can specify the contemplated application so also can one specify an appropriate data structure. In practice it is generally the lack of knowledge of specific applications which renders difficult the choice of data structure. There is no such thing as an optimal all-purpose geographical data structure. Some advocates of particular structures would have one believe in such chimera.

Early attempts at computerized cartography were satisfied to use output equipment as mechanical draftsmen. Figure 1, for example, is a map of the United States which dates from 1963. Although this map contains state boundaries the data set used to create the map, 10400 latitude/longitude pairs in an orderly sequence, does not in fact recognize the states as entities. The first stage of these endeavors had as an objective the ability to draw maps automatically. These maps would then be used in the conventional manner. Efforts in this direction continue, with increasingly intricate maps being produced. The currently available World Data Bank II, for example, contains some 6,000,000 points. Color separation drawings for complete topographical maps, for land use maps, for atlas sheets, and for nautical charts have been produced automatically on at least an experimental basis.

The next natural step is of course to leave the geographical information in the computer, and to never actually produce any illustrative graphic (map), but rather to solve the geographical problem directly, mimicking in the computer what one would do if one had at hand the geographical map. There are some applications in which this objective has been achieved. But generally we are at a stage intermediate to these two steps. Thus we have, interalia, census data collected for states, counties, and other small areal subdivisions. We would like to have these data be associated in the computer with the particular set of locations, i.e. latitudes and longitudes, to which they belong geographically. Thus states, etc. should be recognized as entities in a boundary polygon file. We can then draw maps of the merged files, as in Figure 2. It is fairly apparent that a data structure which is convenient for the mechanical drafting of maps is ackward for the merging of files, or for the recognition of geographical entities. It is often not noticed that the converse proposition also holds. However, as long as the internal geographic representation is "complete", one can convert from one structure to another. Thus, from one point of view, it does not matter which structure is used. The two difficulties are that (a) it is not entirely clear when a geographical data set is "complete", and (b) when the volume of observations is large it may be impractically expensive to convert from one organization to another. The classical cartographic paradigm separates phenomena into classes representable on maps as points, lines, areas, or surfaces. Although somewhat hobbling, this categorization can be used to illustrate the foregoing assertions.

Figure 1: Map of the United States drawn by computer-controlled plotter using geographical detail stored on a magnetic tape released in 1963 by A.V. Hershey.

Point phenomena are easily represented by geographical coordinates. These coordinates are names attached as labels to places on the earth's surface. Since the naming of places is easy one single place may have several different names, or aliases, in different coordinate systems. Two well known systems are latitude/longitude, and transverse Mercator coordinates. But a house, for example, also has an identification using a street name and number, at least in the European culture area. Conversion between these various aliases may be necessary. Early computerized urban transportation studies coded as many as sixteen geographical aliases for each location of interest. But clearly if one knows the latitude and longitude of a house then the county which contains the house could be computed, and need not be recorded. Such redundant coding is used when frequent conversions would otherwise be required.



DEFENSE SPONSORED RESEARCH IN PHYS.,BIOL. AND SOC. SCIENCES (1968)

Figure 2: Merging of locative (geographic) information and substantive data for automatic choropleth map production.

In general the data structure associated with points is relatively simple, and "dot" maps are easily produced automatically. The analysis of point patterns, i.e. the abstract study of the arrangement of dots on a piece of paper, is of importance in biology, geography, archaeology, and several other fields. It is mostly a matter of manipulating the coordinate labels, and should of course be coordinate invariant. Simple point counting within regions occurs in marketing studies and city planning. Often the points have associated with them some other phenomena, e.g., the "point" may be labelled 'New York City', and has a population, etc. Or one may have a time series of spatial point arrangements. These latter data are conveniently represented in the form of a computer movie; Figure 3 shows a frame from an inexpensive movie of this type. The comparable statistical analysis, studying point patterns in space-time becomes very intricate. The simple structure of point observations also suddenly becomes very elaborate when one attempts to establish a relation between the points. If the observational locations are weather stations for example, one might wish to associate points through an "adjacency" relation. Establishing such a relation can be quite intricate; depending on the definition used one may have an NP-complete problem.

Lines are conveniently defined as segments between points (now often called nodes). A coastline, for example, can be described as an ordered set of points, defined by coordinates. But there are many additional entities which are recognized as "lines". A pattern of streets, geologic faults, a stream network, boundaries between regions, and level curves, are all treated within the context of lines. From a drafting point of view these lines have a great deal in common. From an analysis point of view the collection is too heterogeneous to make any sense at all. The study of abstract line patterns seems still to be rather primitive, perhaps similar to the difficulties of studying shapes. Network analysis is perhaps the most advanced. And there are more subtle difficulties. A coastline is not really a well defined entity, but we pretend that it is.

One topic which has been given attention, particularly with respect to lines (although it appears also for points, areas, and surfaces), is map generalization. It is closely akin to the aggregation problem in economics. The cartographic motivation stems from the reduction of detail required when one shrinks a portion of the earth's surface to postage stamp size, i.e., makes a map. Analogies might include the condensed book (textual generalization), the musical overature (acoustical generalization), or cartooning (visual generalization). Yet to be realized are geographical editing systems comparable to the many text editors now available for interactive computing. Interactive graphics is a clear move in this direction.

The "areas" encountered in cartography are bounded patches of the earth's surface, with which are associated phenomena of some sort. The phenomena are often categorical (binary: land versus water; or n-ary: land use type, rock type), occasionally numerical (scalar: taxes contributed, number of people, disease rate; or vectorial: table of the number of interactions with other places; etc.). Generally the patches exhaust the geographical domain of interest. Efforts to date have been directed mainly at the elucidation of methods of coding the boundaries of the patches. Several techniques, some extremely ingeneous, are available. Notable among these systems is one refered to as Dual Independent Map Encoding, which has been used to describe the pattern of streets and enumeration districts for some two hundred U. S. cities (and versions have been exported). In this scheme each street segment is treated as a directed line, with coordinates at the end points, and is associated with an area on the left and an area on the right hand side, and with a street name and with a range of house numbers, Figure 4. One advantage of this scheme is the quasi-automatic topological error checking which it permits. Most recently hierarchical methods and methods permitting nesting of areal phenomena have been described. The hierarchical schemes allow trivial generalization by upward aggregation. The nested schemes recognize,

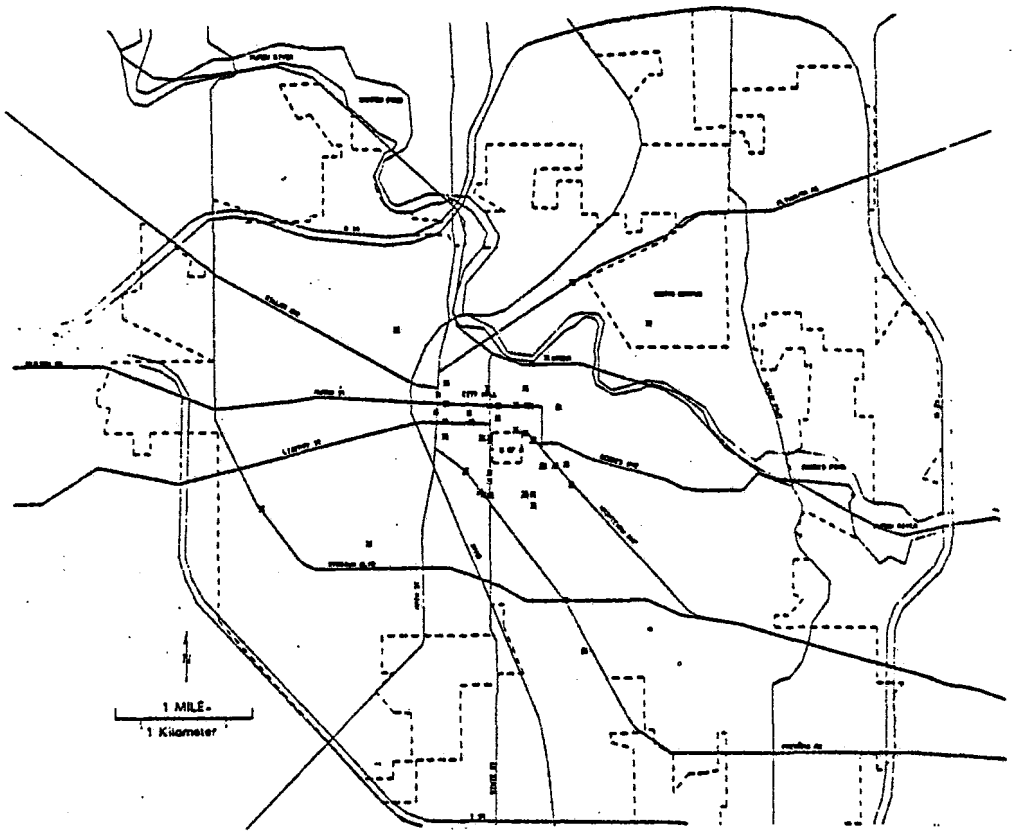


Figure 3: Frame from a computer movie depicting time varying geographic phenomena.(Location of traffic accidents in a city). After Moellering.

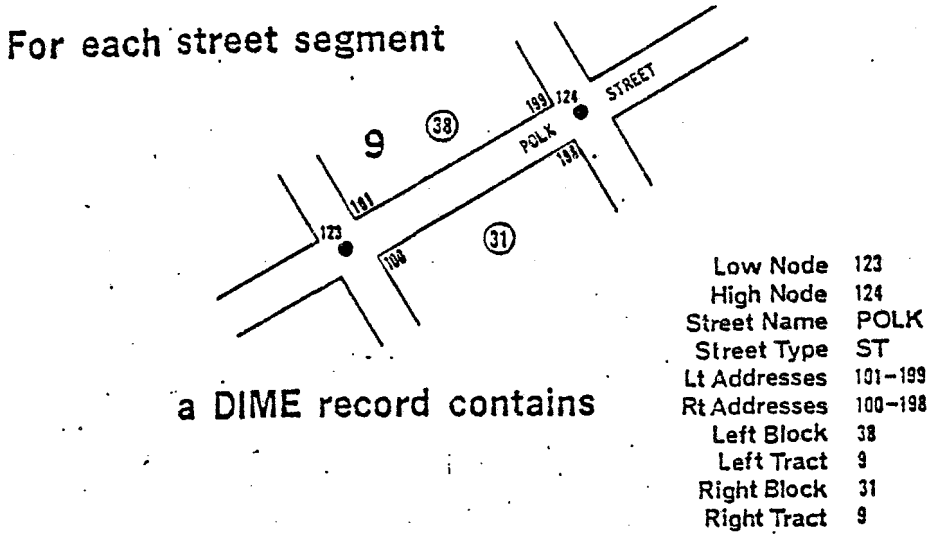


Figure 4: Organization of geographical detail in a DIME file. U.S. Bureau of the Census.

for example, that a tree may be in a park in a city in.... These are probably the most realistic but are not yet widely implemented. All of these schemes describe areas as polygons with a finite number of vertices. They generally allow disjoint pieces and inclusions. They differ mainly in how they arrange the pointers between nodes, arc s, segments, edges, areas and adjacencies. Many workable schemes are now available, for example, to produce maps automatically using the two meters of magnetic tape required to store the 46, 142 point latitude/longitude description of the U.S. counties. Point-in-Polygon routines are commonplace for assigning point phenomena to regions; e.g. converting street address to census tract name. Polygon overlay programs are also available to, for example, convert land use polygons into areal counts by census tract polygons. The cost of this file merging procedure grows as the product of the number of polygons, and is also a function of their complexity. It is thus not an exponential problem but is sufficiently expensive to cause practical difficulties. Regional planning often consists of a concatenation of simple binary decisions which can be caricatured in the following manner: "if land slope is not steep and drainage is good and a road is nearby...then development is permitted." Here the logical intersection of three maps (sets of polygons) is required, and a practical problem might require several dozen such, often repeated as the criterion shift slightly in the process of political compromise. Thus polygon overlay is too expensive in today's primitive computer environment. For analysis purposes other criteria are important. Figure 5, for example, shows a bivariate histogram of U.S. population density by state. It can be thought of as a pictorial representation of a spatial step function, $f(x,y)$. Now a natural way to analyze a function of two space variables is to perform a bivariate spectral decomposition. Geographical central place theory, which predicts a spatial periodicity in the geographical arrangement of population, also suggests such an approach. There are two difficulties in performing such an analysis on these data. Clearly state data are too coarse. A convenient definition of the average spatial resolution of areal data is to take the k^{th} root of the number of observations divided by the measure of the size of the domain, where k is the dimensionality of the domain. In the present instance the square root of the number of contiguous states (48) divided by the land area of the United States yields an average resolution of 250 miles. From the sampling theorem we thus know that we cannot expect to resolve any features in the data which are less than 500 miles in size. Thus this data set is hardly adequate for the study of central places. Conversely, if the mean daily activity field of an individual is 20 miles, then we would need areal units which are one tenth the sizes of counties to be able to say anything meaningful about behaviour of the average individual. Thus for analysis purposes the resolution of the data is very important. To my knowledge the consequences of uneven resolution in a spatial data set have never been examined analytically. The second difficulty in analyzing polygonally defined spatial data is topological. To study the geographical spread of ideas, or of diseases, one needs to know which areas are neighbors of which other areas. This is not terribly difficult to compute. Then one models the process as a relation between neighboring areas. It now becomes very awkward since the number of neighbors can vary a great deal. The difficulty is most easily explained by asking that one devise a set of rules for a chess-like game in which the "squares" on the board have the shape and the continuity relations of the counties of North Carolina. Analysis techniques are usually described in textbooks for spatially homogeneous entities. Thus a question becomes how one does spectral analysis on data packaged in census regions? The natural way to compare two scalar geographical maps is to perform a bivariate cross spectral analysis. How can this be done when the data for the maps is given by two different polygonations?

The obvious answer is to avoid the question, by replacing the polygons by a set of uniform regions, triangles or unit squares. This can be done in either of two ways, though the distinction is rarely noticed in practice. In the first instance one samples at a lattice of points; this is equivalent to multiplication by a Dirac brush, the two dimensional equivalent of the comb function. Alternatively one aggregates data within the cells of a mesh. In the literature these schemes are called "grid" systems, distinguishing them from "polygon" systems. The advantage of a grid system is that the adjacency relation is tooologically constant, a form of spatial stationarity which makes analysis much easier. It also lends itself to bit plane representation in a computer. The disadvantage is

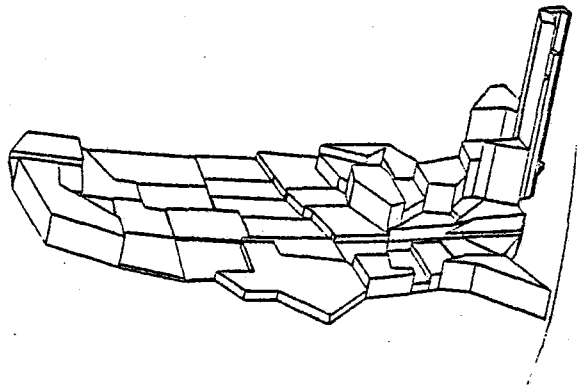


Figure 5: Population density by state for the USA rendered in the form of a bivariate histogram. Program prepared by the author.

that, for a given average resolution, the fit to the data is poorer. These concepts are rarely given precise definition in the literature. For example, one could set up a mean square departure test, keeping average resolution constant, or, conversely, chose the resolution to satisfy some such criterion. At the moment the choices in this area are terribly "practical", that is, atheoretical and ad hoc. There is of course also an obvious relation to storage requirements. Curiously, variable sized triangulations, the natural simplex in two dimensions, and much used in finite element analysis, have not been popular as cells for geographical data storage, probably because automatic scanners cannot easily capture the data in this form. Interconversion between grids using different lattice sizes requires knowledge of the limitations imposed by the sampling theorem, but otherwise causes no great difficulties. Programs also exist to convert polygonal data to grid formats, and the converse.

The typical surface treated in cartography is the topography of the land. Elevations are treated as x,y,z coordinates, either on a uniform lattice in which case the x,y coordinates are implicit, or at critical points, or at randomly sampled points. In all three cases the entire surface is recreated by an interpolation procedure. These interpolation rules are hypotheses about nature, which may or may not be true. Interpolation invariably requires that one decide which observations are adjacent to each other--all points may be considered neighbors, but then some are more neighborly (have greater influence) than others. Display formats include block diagrams and level curves or contours. For some applications it is advantageous to store the contours as lines (strings of coordinates) and the topological nesting tree. Some difficulties are encountered when other files, streams or roads, are combined with topographical surfaces unless the two were collected simultaneously since they otherwise do not exactly match.

BIBLIOGRAPHY

R. Aangeenbrug, Auto-Carto II, Proc. Int. Conf. on Comp. Asst. Cartography, ACM, Reston, Va, 1975.

E. Amidon, G. Aiken, "Algorithmic Selection of the Best Method for Compressing Map Data Strings", Comm. ACM , 14 (1970), pp. 769-774.

H. Bailey, "Two Grid Systems that Divide the Entire Surface of the Earth into Quadrilaterals of Equal Area", Transactions, Am. Geophysical Union, XXXVII (1956), pp. 628-635.

R. Barraclough, "Geographic Coding", pp. 219-296 of Federal Statistics, Presidents Commission, Washington DC, Government Printing Office, 1971.

M. Bartlett, The Statistical Analysis of Spatial Pattern, London, Chapman & Hall, 1975.

R. Batson, et al, "Computer Generated Shaded Relief Images", Journal of Research U.S. Geological Survey, 3, 4 (July 1975) 401-408.

R. Baxter, Computer and Statistical Techniques for Planners, London, Methuen, 1976.

H. Blum, "Biological Shape and Visual Science", J. Theor. Bio., 38 (1973), pp. 205-287.

E. Burr, "Sharpening of Observational Data in Two Dimensions", Australian Journal of Physics, 8 (1955), 30-53.

W. Burton, Representation of Many-Sided polygons and polygonal lines for rapid processing", Comm. ACM, 20 (1977), p. 166-171.

K. Clayton, "Geographical Reference Systems", The Geographical Journal, 137, 1 (1971), pp. 1-13.

A. Cliff, et al, Elements of Spatial Structure, Cambridge, University Press, 1975.

D. Connelly, "The Coding and Storage of Terrain Height Data: An Introduction to Numerical Cartography", M.S. Thesis, Ithaca Cornell University, 1968.

M. Dacey, "Some Questions about Spatial Distributions", pp. 127-152 of R. Chorley, ed., Directions in Geography, London, Methuen, 1973.

G. Deecker, and J. Penny, "On Interactive Map Storage and Retrieval", Can. J. of Operational Research and Information Processing, 10 (1972), pp. 62-74.

P. Delfiner, and J. Delhomme, "Optimum Interpolation by Kriging", pp. 96-114 of J. Davis, and M. McCullagh, eds., Display and Analysis of Spatial Data, New York, J. Wiley, 1975.

K. Deuker, "A Framework for Encoding Spatial Data", Geogr. Anal., 4 (1972), pp. 98-105.

D. Douglas, T. Peucker, "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature", The Canadian Cartographer, x, 2 9o973), pp. 112-122.

E. Epstein, "Stochastic Dynamic Prediction", Tellus, 21, 6, (1969), pp. 739-759.

H. Ferguson, "Point in Polygon Algorithms", Seattle, Urban Data Center, 1974.

H. Freeman, "On the encoding of arbitrary geometric configurations", IRE Trans. Elec. Comp., 10 (1961), pp. 260-268.

H. Freeman, "Computer Processing of Line-Drawing Images", ACM Computing Surveys, 6, 1, (March 1974), pp. 57-97.

L. Gale, "Recommendations on the type of coordinate system for correlating files in a data bank system", The Canadian Surveyor, Vol. 23, 1969.

J. Gardner, A Study of Environmental Monitoring and Information Systems, Iowa City, Institute of Urban and Regional Research, 1972.

General Services Administration, Worldwide Geographical Location Codes, Washington DC, Government Printing Office, 1972.

R. Gilchrist, and G. Cressman, "An Experiment in Objective Analysis", Tellus, 6, 4 (1954), pp. 309-318.

S. Guptil, Spatial Filtering of Nominal Data: An Exploration, Ph.D. Thesis Department of Geography, University of Michigan, Ann Arbor, 1975.

T. Hagerstrand, "A Monte Carlo Simulation of the Diffusion of Innovations", in B. Berry, and D. Marble, eds., Spatial Analysis, Englewood Cliffs, Prentice Hall, 1968.

F. Hearle, and R. Mason, A Data Processing System for State and Local Governments, Englewood Cliffs, Prentice Hall, 1963.

J. Holloway, "Smoothing and Filtering of Time Series and Space Fields", Advances in Geophysics, 4 (1958), 351-389.

C. Huijbregts, "Regionalized Variables and Quantitative Analysis of Spatial Data", pp. 38-53 of J. Davis, and M. McCullagh, eds., Display and Analysis of Spatial Data, New York, J. Wiley, 1975.

J. Jacobsen, "Geometric Relationships for Retrieval of Geographic Information", IBM Systems Journal, Vol. 7, Nos. 3 and 4, 1968, pp. 331-341.

W. Kaula, "Theory of Statistical Analysis of Data Distributed over a sphere", Reviews of Geophysics, V, 1 (1967), p. 83-107.

K. Kraus, and E. Mikhail, "Linear Least Squares Interpolation", Photogrammetric Engineering, 1972, pp. 1016-1029.

M. Kriger, "Sugar: A high level programming Language for Geographical Analysis and Mapping", Computer Graphics, 10, 1 (1976).

B. Kubert, et al, "The Perspective Representation of Functions of Two Variables", Journal, ACM, 15, 2 (April 1968), pp. 193-204.

P. Lewis, Maps and Statistics, New York, Wiley, 1977.

R. Loomis, "Boundary Networks", Comm ACM 8 (1965), pp. 44-48.

E. MacDougall, "Optimal Generalization of Mosaic Maps", Geographical Analysis, 4, 4 (1972), 417-423.

D. Maling, Coordinate Systems and Map Projections, London, G. Philip, 1973.

B. Mandelbrot, Fractals, San Francisco, Freeman, 1977.

D. Mark, "Computer Analysis of topography: A Comparison of terrain storage methods", Geografiska Amaler, 57A, 3-4 (1975).

G. Matheron, The Theory of Regionalized Variables and its Applications, Cahiers du Centre de Morphologie Mathematique de Fontainebleau, No. 5, Ecole Superieure des Mines, 1971.

R. Merrill, "Representation of Contours and Regions for Efficient Computer Search", Comm. ACM, 16, 2 (1973), pp. 69-82.

H. Moellering, "Mapping Traffic Crashes in Washtenaw County", Ann Arbor, Highway Safety Research Institute, 1973, pp. 1-5.

H. Moritz, Eine Allgemeine Theorie der Verarbeitung von Schweremessungen nach kleinsten Quadraten, Heft Nr. 67A, Munchen, Deutsche Geodatische Kommission, 1970.

S. Morse, "A Mathematical Model for the Analysis of Contouring Line Data", J.A.C.M., 15, 2 (1968), pp. 205-220.

D. Moyer, and K. Fisher, Land Parcel Identifiers for Information Systems, American Bar Foundation, 1973.

S. Nordbeck, and B. Rystedt, "Isarithmic Maps and the Continuity of Reference Interval Functions", Geografiska Annaley, 52 B, 2 (1970) pp. 92-123.

S. Nordbeck, and B. Rystedt, Computer Cartography, Lund, Bloms Bokfryckeri, 1972.

J. Palmer, "Computer Science Aspects of the Mapping Problem", pp. 155-172 of J. Davis, and M. McCullagh, Display and Analysis of Spatial Data, New York, J. Wiley, 1975.

T. Peucker, Computer Cartography, College Resource Paper 17, Washington, DC, Assn. Am. Geographers, 1972.

T. Peucker, N. Chrisman, "Cartographic Data Structures", The American Cartographer, 2, 1 (P 75), pp. 55-69.

J. Pfaltz, A. Rosenfeld, "Computer representation of Planar Regions by their skeletons", Comm. ACM, 10 (1967), pp. 119-125.

R. Phillips, Computer Graphics in Urban and Environmental Systems, Proc. IEEE, 62, 4 (1974) pp. 437-452.

E. Pielou, An Introduction to Mathematical Ecology, New York, Wiley, 1969.

J. Rayner, "The Time Element in Spatial Process", pp. 179-216 of D. Deskins, ed., Geographic Humanism, Analysis, and Social Action, Mich. Geogr. Pub. 17, Ann Arbor, Dept. of Geogr., Univ. of Michigan, 1977.

J. Rayner, An Introduction to Spectral Analysis, London, Pion, 1971.

D. Rhynsburger, "Analytic Delineation of Thiessen Polygons", Geographical Analysis, 2 (1973), pp. 133-144.

A. Rogers, Statistical Analysis of Spatial Dispersion, London, Pion, 1974.

A. Rosenfeld, J. Pfaltz, "Sequential Operations in Digital Picture Processing", J.ACM, 13 (1966), pp. 471-494.

A. Schmidt, ed., "Topological Data Structures for Geographic Information Systems: A Symposium", to appear.

M. Shamos, Computational Geometry New York, Springer, 1977.

J. Sibert, Spatial Autocorrelation and the Optimal Prediction of Assessed Values, Mich. Geogr. Pub. 14, Ann Arbor, Dept. of Geography, Univ. of Michigan, 1975.

R. Sokal, and K. Gabriel, "A New Statistical Approach to Geographic Variation Analysis", Systematic Zoology, 18, 3 (1969), 259-278.

L. Stegena, "Tool for the Automation of Map Generalization", pp. 66-95 of E. Csati, ed., Automation: the New Trend in Cartography, Budapest, Int. Cart. Assn., 1974.

I. Sutherland, et al, "A Characterization of Ten Hidden Surface Algorithms", ACM Computing Surveys, 6, 1 (March 1974), pp. 1-56.

P. Switzer, "Estimation of the Accuracy of Qualitative Maps", pp. 1-13 of J. Davis, and McCullagh, Display and Analysis of Spatial Data, New York, J. Wiley, 1975.

P. Switzer, "Mapping a geographically correlated Environment", pp. 235-269 of G. Patil, et al., eds., Statistical Ecology, Vol. 1, University Park, Penn. State University Press, 1971.

W. Tobler, "Automation and Cartography", The Geographical Review, XLIX, 4 (1959), 526-534.

W. Tobler, "Geographical Filters and their Inverses", Geographical Analysis, I, 3 (1969), 234-253.

W. Tobler, "Linear Operators applied to Areal Data", pp. 14-37 of J. Davis, and M. McCullagh, Display and Analysis of Spatial Data, New York, J. Wiley, 1975.

R. Tomlinson, ed., Geographic Data Handling, Vol. I and II, Ottawa, IGU Commission, 1972.

R. Tomlinson, et al, Computer Handling of Geographic Data, Paris, UNESCO, 1976.

U.S. Department of Commerce, Bureau of the Census, "Documentation for the GBF/DIME file, GEO-202, Revised April 1976.

T. Waugh, D. Taylor, "An Example of an Operational System for Computer Cartography" Canad. Cartographer, 13 (1976), pp. 158-166.

P. Werner, "National Geocoding", Annals, Assn. Am. Geographers, 64, 2 (1974), 310-317.

P. Yoeli, Analytical Hill Shading, Surveying and Mapping, 25, 4 (1965), 573-579.

WORKSHOP 5

STATISTICAL COMPUTING IN ADVERSARY PROCEEDINGS

Chair: John S. de Cani, University of Pennsylvania

THE LAWYER AND THE STATISTICAL--COMPUTER EXPERT

Alfred A. Porro, Jr.
Law Offices
Porro Building, Suite 100
10 Stuyvesant Avenue
Lyndhurst, N.J. 07071

ABSTRACT

The contemporary search for truth and persuasion in adversary legal proceedings has given birth to a new aspect of the creature known as Jurimetrics. The melding of the evidential rules and methods of the Court with the science and technology of measurement is the challenge. This marriage has forced to the surface the need for the various professions involved to understand the basic concepts of the other. The author provides the scientist and technological expert in the statistical and computer fields with a legal prospective. The need for increased qualifications of all involved, the necessity to progress toward uniform standards and the ability to communicate is demonstrated.

TABLE OF CONTENTS

I. "So Help you God"

A. The Scene: Law, Science and Technology in Court.

The gavel goes down, the witness is called and the first response elicited: "Do you swear to tell the truth, the whole truth and nothing but the truth, so help you God". More and more scientific and technological experts are declaring the affirmative pledge each judicial day of our contemporary lives. The significance of the closing phrase,"so help you God", can only be totally appreciated by one who has the insight of personal experience. That is to say--from either the lawyers or experts standpoint! This transcendental experience raises each beyond their prior realm of understanding to a new level of communication. Hopefully, the result will be to stimulate the intuition of the decision maker to the point of being part of the process that is occurring. When reaching this horizon the burden of proof will be carried.

The trial attorney is educated and trained in an atmosphere of adversariness and persuasion. He utilizes a set of rules and their exceptions as a tool. With this back drop he steps into a maze of foreign terminology, concepts, methods and conclusions. On the other hand, the expert bedded in a world of testing hypothesis, methodology and measurements of probability is riddled with the obstacles of the judicial course. This merger can be a confluence process which will enlighten and serve as a valuable vehicle of proof and realization. On the other hand, it can be a disasterous collision. The essence of the former is a matter of qualifications, striving for uniform standards and an ability to communicate.

Since trials by ordeal the adversary system has sought to utilize the impact of science and technology to prove its case. Richardson, James R., Modern Scientific Evidence, Section 1.6, Page 7 et seq. (W. H. Anderson Co., 1961). From the biblical writings to the present theological rebirth, the application of a jurisprudence has seldom been without the technical expert. This contemporary world is maximizing upon the prior increasing emphasis given to scientific and technological proofs.

In consideration of the countless data, methods and evidence provided by the statistical and computer fields--in penance and apology for the misuse and abuse that some have put them to--the following is an attempt to sensitize scientific and technological experts to the legal backdrop. This sensitivity has progressed from need to an essential element. It is an invitation to join in an effort to come closer to a full insight.

B. The Subject: A Matter of Expertise.

Not all subjects are legally open to the application of the rules governing the expert testimony. Unlike the layman, the expert may express opinions relative to matters not observed.[1] The essence of expert testimony is in the form of opinions. However, throughout the decades various restrictive standards have been first applied with respect to the subject matter covered by the experts testimony. It has been deemed to be necessary to be beyond the general knowledge of the layman.[2] It has been declared that it must be necessary that the expert testimony be helpful to the trier of fact in determining the truth.[3] Rule 72 of the Federal Rules of Evidence requires that the Court determine that the subject is a proper one for expert testimony. Generally such testimony will not be permitted unless the matter involves "scientific, technical or other specialized knowledge" beyond common knowledge.[4] Much discretion is vested in the trial judge in this respect. The subject matter must not only be helpful in understanding the evidence and determining the fact in question but also must be legally relevant.[5] An important judicial standard often not brought to the attention of the expert is that of the doctrine of "acceptability". Ironically, this doctrine requires that the expert evidence "be evaluated on the basis of its acceptability to those of acknowledged expertise in the relevant area".[6] It must lie beyond the realm of untested hypothesis and scientific speculation. The key legal standard is expressed in Frye v. United States, 293 Fed. 1013, 1014 (CA DC 1923). There the Court stated: "Just when a scientific principal or discovery crosses the line between the experimental and demonstrable stages is difficult to define. Somewhere in this twilight zone the evidential force of the principal must be recognized, and while Courts will go along way in admitting expert testimony deduced from a well recognized scientific principal or discovery, the thing from which the deduction is made must be sufficiently established to have gained general acceptance in the particular field in which it rests."

On the other hand, more recent decisions appear to give more leeway in the admissibility of such evidence, even if there is disagreement technically regarding its accuracy, where the conclusions are relevant and supported by a qualified expert witness. In

such instances the objections to its acceptance will go to the weight rather than the admissibility of the evidence.[7]

As the ingeniousness of the world of science and technology grows, so does the scope of its impact upon judicial proceedings.

C. The Expert: Qualification and Presentation.

An expert must have "something different to contribute".[8] He must have qualification by "knowledge, skill, experience, training or education".[9] Thus, the expert's academic background and experience surface as being elementary to the injection of opinion into the evidential stream. The rules of qualification are not solidified in cement and much is left to the discretion of the Court. In most instances the expert will qualify "because they have specialized skill or training which enables them to perceive and interpret events in ways that ordinary lay people cannot * * * Courts look at experts from a functional prospective. If the Court is satisfied that an individual has the ability to draw inferences which the ordinary juror could not draw and that these inferences will aid the jury in rationally evaluating a fact in issue, the individual may qualify as an expert".[10]

The experts qualifications often go beyond allowing opinion testimony. They may be needed to provide valuable specialized information on a given subject matter. Such evidence is informational and factual, although requiring expert qualification.

Though an expert may qualify to give testimony this does not exempt attack by way of cross examination as to credibility. Admissible evidence is always subject to the cross examination aimed at limiting the weight to be given to it.

The presentation of expert testimony follows two basic categories:

1. Personal knowledge
2. Hypothetical in nature

The first category follows much the same procedure as that of lay testimony, excepting it is based upon specialized expertise. An expert may testify to relevant facts gained through personal observation. These observations must -- normally be set forth before expert opinion is allowed to be elicited -thus-----the observations lay a factual basis. Opinions based on hypothetical facts, on the other hand, are also commonly utilized. An expert may not have actual knowledge of some or all of the facts necessary to support an opinion; these facts may be set forth hypothetically provided there is independent evidence which has been introduced in trial to support them.

II. The Statistician--Computer Type.

A. General Trend: A Legal Prospective.

Jurimetrics is a term that has been coined to characterize "the scientific investigation of legal problems".[11] For more than a quarter of a century the growing impact of mathematical measurements and techniques has been slowly and continuously invading the legal profession and its call. The computers invasion of the law practice has been deemed a "professional revolution that will be largely complete by 1990".[12] In the law office it has established itself as being invaluable for word processing and accounting and legal research. One of the most recent implications of computer technology is in that of litigation support and the management of documents and information for litigation.[13] There has also been much debate regarding the use of probability theory to help resolve problems of proof in actual litigative proceedings.[14] To date there is little uniformity regarding utilization of mathematical evidence, particularly where used as a probabilistic proof to establish identification for fact. Although scientific proofs have been consistently used to identify persons, such as through fingerprints, photographs and measurements, less attention has been given to the theory of probabilities as applied to such questions of identification.[15] In the case of People v. Collins, 68 Cal. 2d, 319, 66 Cal Rptr. 497, 438 P. 2d 33 (Sup. Ct., 1978) the use of mathematical probability to establish an identification was condemned as the Court said: "While we discern no inherent incapability between the disciplines of law and mathematics and intend no general disapproval or disparagement of the latter as a auxiliary in the fact--finding process of the former, we cannot uphold the technique employed in the instant case * * * the testimony as to mathematical probability infected the case with fatal error and distorted the jury's traditional role of determining guilt or innocence according to long-settled rules. Mathematics, veritable sorcerer in our computerized society, while assisting the trier of fact in the search for truth, must not cast a spell over him. We conclude that on the record before us defendant should not have had

his guilt determined by the odds and that he is entitled to a new trial."

This was the reaction of one Court where a prosecutor called an instructor of mathematics at a State College to attempt to bolster identifications of the defendant. The facts disclosed a robbery committed by a Caucasian woman with a blonde ponytail, who left the scene accompanied by a Negro with a beard and mustache. Through the "product rule", i.e. that the probability of the joint occurrence of a number of mutually independent events is equal to the product of the individual probabilities that each of the events will occur, the opinions of the expert were received. The Court noted that the testimony lacked an adequate foundation both in evidence and in statistical theory and that the testimony and the manner in which the prosecution used it distracted the jury from its proper and requisite function of weighing the evidence on the issue of guilt. The case demonstrated failure on behalf of the prosecutor to present any statistical evidence whatsoever in support of the probabilities for the factors selected and on the other hand the ineffectiveness of defense counsel apparently "unschooled in mathematical refinements".[16] The Collins case raises haunting issues and considerations which underlie many of the Court decisions involving the statistical--computer types--degree of reliability and qualification and the policy issues of usurpation and prejudice. Thus, the State of the Art appears to be progressively &rapidly gaining in its invasion of the legal profession, however, its progress is somewhat slower and more cautious when applied to the legal reception of statistical or computer evidence.

B. Computer Generated Evidence.

The development of the reported cases dealing with the law regarding admissibility of computer generated evidence stems initially from the subject of business records kept by computer. This is to be distinguished from computer-printouts prepared especially for litigation. Start with---- Transport Indem. Co. v. Seib, 178 Neb. 253, 132 N. W. 2d (1965) where, a computer print-out of accounts receivable was admitted as evidence after testimony by the insurer's director of accounting established that the computer records were made in the usual course of business, identified the records, explained all entries on the print-out, and described in detail the procedure for using the computer. 132 N. W. 2d at 874-75. The Court held that the fact that this particular print-out was made specifically for trial was a distinction without merit. Id.

In Merrick v. United States Rubber Co., 7 Ariz. App. 433, 440 P. 2d 314 (1968), a computer print-out of records of consigned accounts was admitted as evidence testimony of the company's credit officer verified the normal method of keeping the records and that he was personally familiar with the account in question. The court found this to be a sufficient foundation for the reception of the print-out despite the fact that the witnesses had no personal knowledge of the computer's operation. 440 P. 2d at 317.

In King v. State ex rel. Murdock Acceptance Corp., 222 So. 2d 393 (Miss. 1969) the Court laid down the following criteria for the admissibility of computerized business records: "That the electronic computing equipment is recognized as standard equipment; the entries are made in the regular course of business at or reasonably near the time of the happening of the event recorded, and the foundation testimony satisfies the Court that the sources of information, method and time of preparation were such as to indicate its trustworthiness and justify its admission." 222 So. 2d at 398. See also United States v. De Georgia, 420 F. 2d 889 (9th Cir. 1969); D & H Auto Parts, Inc. v. Ford Marketing Corp., 57 F R D 548 (F.D.N.Y. 1973); Union Electric Co. v. Mansion House Center North Redev. Co., 494 S. W. 2d 309 (1973); Nelson Weaver Mortgage Co. v. Dover Elevator Co., 283 Ala. 324, 216 So. 2d 716 (1968).

Sears Roebuck and Co., v. Merla, 142 N.J. Super. 205, 361 A. 2d 68 (App. Div. 1976) was an action on a book account. The trial court dismissed the complaint inter alia because it would not accept a computer print-out of the transactions in question. The Appellate Division reversed, holding in pertinent part: " . . as long as a proper foundation is laid, a computer print-out is admissible on the same basis as any other business record. Computerized bookkeeping has become commonplace. Because the business records exception is intended to bring the realities of the business world into the courtroom, a record kept on computers in the ordinary course of business qualifies as competent evidence." Cf. State v. Hibbs, 123 N.J. Super 12, 301 A. 2d 789 (Mercer Cty. Ct. 1972), affirmed, 123 N.J. Super. 124, 301 A. 2d 775 (App. Div. 1973) (computerized electronic telephone tracing equipment). See generally N.J. R.Evid. 1 (13) and 63 (13) (read together clearly

allow computerized business records).

Perhaps the most recent decision in point is United States v. Scholle, 553 F. 2d 1109 (8th Cir. 1977). Therein, the prosecution introduced a computer read-out prepared by the Drug Enforcement System Administration. This system computerizes the physical characteristics of drugs seized and tested around the country and is used to alert law enforcement officials of the new trends, new drugs, etc. This particular print-out showed a chemical composition of cocaine similar to that which defendant was charged with having possessed and sold. In affirming the conviction the court held inter alia: "The determination of relevancy concerning the computer evidence was within the broad discretion of the trial judge. The complex nature of computer storage calls for a more comprehensive foundation. Assuming properly functioning equipment is used, . . . the original source of the computer program must be delineated, and the procedure for imput control including tests used to assure accuracy and reliability must be presented." 553 F. 2d at 1124-25 (citations omitted).

As the cases above illustrate, computer generated evidence should be admissible if a proper foundation is laid. One commentator has stated: "Admissibility should depend on whether the computer generated data is related to a material issue in the case; whether the data fits within a chain of inferences which would connect the data to an ultimate issue in the case; and whether the data can and will be presented in such a manner that it will not unduly confuse the issues or so confound and overawe the jury. . . that it would be prejudicial to the opposing side. Jenkins, Computer--Generated Evidence Specially Prepared for Usé at Trial," 52 Chicago-Kent L. Rev. 600,608-09 (1976).

Another author, speaking of the credibility of such evidence, has stated: "To insure the credibility of the information, it must be demonstrated that the programs are functioning properly and are free from errors. It must also be shown that no unauthorized programs were permitted to control the computer, since such programs could have altered the information. It must be demonstrated that unauthorized persons . . . . . could not gain access to the computer . . . Finally, it must be established that the information actually presented to the trier of fact is an accurate copy of the computer-stored information, and that the information should not be accepted simply because it is presented in the form of an authoritative-looking computer print-out." Sprowl, "Evaluating the Credibility of Computer-Generated Evidence," 52 Chicago-Kent L. Rev. 547, 557-58 (1976).

Another commentator has broken down the data processing system into five stages: 1. generation and assembly; 2. input; 3. operation; 4. storage and retrieval; 5. output. Roberts, "A Practitioner's Primer on Computer-Generated Evidence", 41 U. Chi. L. Rev. 254, 264 (1974). For each of these stages, he proposed a series of questions which counsel and the expert should consider in preparing for the introduction of computer evidence. Thus, important questions to be asked with regard to the generation and assembly stage include:

1. What is the true source of the data?
2. What processes do the original data pass through in the generation and assembly stage before arriving at the input stage?
3. What steps are taken to detect error in the original data and prevent the introduction of error into, or loss of, that data as it passes through the generation and assembly stage?
4. What hearsay or best evidence problems, if any, are suggested by the underlying data?
5. What has been the prior experience with the data collection and assembly procedures?

Id. at 264-65. As to the second stage (input), the following questions might be considered:

1. What types of input devises are used?
2. What procedures are used for error detection?
3. Is the input operation performed in multiple locations?

Id. at 266-67. As to the third stage (operation), Roberts asks the following:

1. Is the CPU (Central Processing Unit) and related equipment, and the configuration in which they are being used, appropriate for the application?
2. Are the personnel. . . competent to operate that equipment . . .?
3. Is there adequate documentation?
4. Do the programs do what they are supposed to do?

Id. at 267-69. In stage four (storage and retrieval), the following questions are asked:

1. Are the storage and retrieval procedures adequate to insure that data will not be lost, changed, or incompletely retrieved and that erroneous data will not be retrieved?
2. Are there adequate safe-guards to prevent tampering with the data?

Id. at 270. Finally, the following questions are asked as to the fifth stage (output):

1. Does the outputting program cause the computer to generate all relevant information in a valid format?
2. Does the out put generated for evidentiary purposes differ from the output generated for normal business use?
3. For what purpose is the output used in the normal course of business?
4. For what period of time has been the output been relied on and in what form?
Id. at 271-72.

Given this secondary authority, the counsel and expert should experience little difficulty in having computer-generated evidence admitted, providing that satisfactory answers to the questions are provided to the trial court.

The cases dealing with computer generated data prepared specifically for trial are scarce indeed. In United States v. Dioguard; 448 F. 2d 1033 (2d. Cir. 1970), defendant was charged with fraudulently concealing assets from the trustee in bankruptcy. The prosecution made a computer run of the company's inventory, sales and purchases. Without any discussion as to "why", the appellate court followed the trial court's decision to admit such evidence. Indeed, rather than commenting on whether computer evidence prepared for trial was admissible or not (apparently assuming that it was), the court took the government to task for not supplying copies of the print-out to the defense.

Despite this lack of direct precedent, one commentator has expressed the following rationale for admission of evidence prepared for trial:

". . . Models . . . may be admissible as expert opinion evidence. The oral testimony of an expert could be based on a computer study or model which had processed the sort of evidence typically considered by the expert in his field or it could be typical for the expert in the particular field to rely on just such a model as happens to have been constructed."

Jenkins, "Computer-Generated Evidence Specially Prepared for Use at Trial," 52 Chicago-Kent L. Rev. 600, 607 (1976). This would be no problem under Fed. R. Evid. 703: "If of a type reasonably relied on by experts in the particular field in forming opinions or inferences upon the subject, the facts or data need not be admissible in evidence."

On the other hand, N.J.R. Evid. 56 (2)would appear to be somewhat more restrictive, by requiring that expert opinion must be based on facts or data established by the evidence at trial. Thus, in New Jersey State Court, it would appear that the computer printouts themselves would have to be admissible before an expert could----extrapolate the data and give his opinion thereon.

The precedent, a term of art in the world of jurisprudence, clearly sets forth certain basic standards which the attorney and experts alike should be aware of. Computer-generator evidence has established its precedent in law. However it has not reached its potential nor full extent of creative presentation. Utilizing the precedent the synergistic effect of communication between a knowledgeable attorney and qualified expert leaves a wide span of potential. The proper legal foundations laid[17] combined with the process of jurimetric preparation, new horizons in evidential receptions will continually be reached. The development of the cases dealing with computer generated evidence clearly demonstrate a trend in this area -- progressing toward better qualification, closer unification of standards and improvement of communication. Substantial progress has been made and continues to be made.

C. The Statistician: Most Probably Misunderstood.

Of all of the scientific and technical experts the statistician is one of the least understood and most probably feared in the Courtroom. Until recently he was not an often visitor there, and perhaps, it is because the legal mind and training is quite foreign indeed from that of the statistician. Anyone having experience with the legal system soon comes to the realization that exactness and accuracy is not necessarily its gravamen. Thus, for lawyers and judges alike, the statistician is indeed difficult to understand. The lack of qualification in this respect from the lawyers and judges standpoint magnifies the element of failure to be able to communicate.

The role of the statistician is also seriously misunderstood by the Court. Putting aside the controversial issue of using this statistical theory to establish the relevancy of evidence itself;[18] this expert can provide extremely valuable direct evidence to aid the Court in determination of the truth. However, as demonstrated by the Collins case, his function stimulates a fear of usurpation of the role of the fact finder. This fear is primarily the result of the three fold pit falls of lack of qualification, non existence of uniform standards and inability to communicate. Statisticians formula and methodologies are beyond the qualification & scope of the judge. There are no legal uniform standards to be applied. Finally, an inability of the two diverse fields to communicate magnifies the problem and results in this apprehension. Statisticians conclusions tend to go to the core of the issue at stake, identity, probability of truth or likelihood of accuracy. However, this testimony, as any other testimony, should be viewed as an aid to the Court and not as a usurpation. Testimony is not submitted nor intended to be conclusive as to the issue in question. Indeed, to exclude it because of the aforementioned three fold pitfalls is indeed a misjustice. To fear that it will overwhelm the fact finder is to emphasize the threefold hazard discussed.

Rather than discuss the role of the statistician in the abstract a particular framework will be helpful. Since the essence of this experts work deals with probability, discriminations, random sampling and related statistical conclusions, the subject of jury selection and composition would appear to be of particular relevance. In recent years the legal profession has become aware of the inherent importance of jury selection procedures and composition to the doctrine of due process.[19] It is now axiomatic that neither a petit or grand jury can be discriminatorily selected.[20] A litigant is entitled to a jury selected from a pool which is representative of a valid cross section of the community.[21] At the Federal lev l, the Jury Selection and Service Act requires: "Grand and Petit Juries selected at random from a cross section of the community".[22] Most states have similar requirements.[23] Thus, there is no tolerance for a system that automatically excludes any identifiable class of persons.[24] Further, the Constitution also protects a litigant against the system that will result in an underrepresentation of any such groups, even if it is not the result of an automatic exclusion.[25] It is immaterial whether or not actual discrimination occurred[26]; nor is it relevant that the failure of the existence of the cross section was the result of good faith or lacked any improper motive.[27]

Thus, the role of the statistician in this type of litigation can be clearly seen from the outset. He can serve as a valuable tool for the Court to determine whether or not discrimination exists, whether a representative cross section exists and whether or not the procedures utilized constituted a random sampling. This initial role can be performed by no other profession as competently and qualifiedly as that of the statistician.

A second role for the statistician exists due to-- the development of the law on this subject. A key phrase with great legal significance provides the key, i.e. "cognizable class". For example, the law readily recognizes race[28], sex[29] and economic status[30] as some of these classifications. Likewise, certain cognizable groups have been permitted to be exempt specifically by statute and upheld as necessary and reasonable.[31] Various tests have been formulated to set forth legal criteria for establishing a cognizable class. Some of these tests have been yielding varying and non uniform standards because of the lack of qualification of the lawyers and the Courts and a failure to communicate with the statistician. Typical of these tests is that set forth in the case of United States v. Gusman, 337, Fed. Sup. 140,143-144 (1972): "A group to be cognizable for present purposes must have a definite composition. That is, there must be some factor which defines and limits the group. A cognizable group is not one whose membership shifts from day to day or whose members can be arbitrarily selected. Secondly, the group must have cohesion. There must be a common thread which runs through the group, a basic similiarity in attitudes or ideas or experience which is present in members of the group and which cannot be adequately represented if the group is excluded from the jury selection process. Finally, there must be a possibility that exclusion of the group will result in partiality or bias on the part of juries hearing cases in which group members are involved. That is, the group must have a community of interests which cannot be adequately protected by the rest of the populace." Under such a criteria the Courts in New York State, where the test, gave birth have recognized "students" as a cognizable group.[32] Although

other States have ended up with the same result,[33] one has- managed to end up with differing results within the same State utilizing this same criteria.[34] The application of such criteria should be uniform and has been modified and criticized.[35] The mathematical method of analysis does provide the Court with an accurate and reasonable uniform standard for determining such cognizable groups as a matter of law.[36] However, the need for qualification of lawyers and judges to be able to comprehend is essential and the ability of both the expert and the legal type to be able to communicate is the next step.

A third area in which the statisician can be of invaluable aid to the Court in jury selection and composition is that of providing evidence of a statistical "significance" between the resulting composition of the jury pool and a cognizable group in the population. Such proof can establish a prima facia showing of discrimination.[37] In both the Turner and Sim cases the United States Supreme Court struck down selection processes wherein the showing by a defendant was that a statistical disparity existed. Thus, once a party is able to make a showing that there is a statistical "significance" between the jury wheel or pool and the population, even in just one category, a prima facia showing of discrimination is made. In some instances the statistician may want to go further and show why! However, that is not essential for purposes of establishing a prima facia case. It is recommended strongly where such a possibility exists. For example, failure of the jury selector to keep updated voter registration roles may be a statistical cause of the discrepancy; this is powerful evidence and should be presented.

The three major areas of testimony of the statistician in this type of case require extensive preparation. In order for his product to be admissible and credible it must be rooted in a solid foundation,--statistically and legally.[38] Such studies involve a number of crucial steps. Initially, accurate data must be compiled relative to the characteristics of the population of the area--demographic data. The datum must be reliably and accurately collected. Secondly, data as to the demographic characteristics of the persons appearing on the qualified jury list must be selected. This information most often can be obtained from the juror qualification forms that have been returned to the Court by the persons. From the data supplied, percentages are computed. Figures and the percentagesare then put into a formula which has been computerized. The computer will supply the results of the computations which then must be interpreted by the expert. A formula often utilized to analyze the data is set forth in detail on page 83 of Doctor Sperich's article entitled "Statistical Decision Theory in the Selection of Grand Jurors", Supra". This formula has acceptance in the scientific community for purposes of computing probabilities, including the probability of discrimination in jury selection. The result of using such a formula to analyze the data is to produce computations to enable the statistician to testify whether or not there was discrimination in the jury selection process. The information collected is related to statistical sampling theory. If a true random selection procedure was utilized to constitute the jury pool, this would mean that each relevant individual in the population from which the pool is to be drawn would have exactly the same chance of being selected as any other person in the population. No one would have a greater chance. No one would be favored with either a greater or lesser chance. Thus, there would be no legal or statistical discrimination. Depending on the Court's qualification and subjective influences, combined with the experts ability to communicate, a valuable aid is provided to the Court in deriving at its decision. No usurpation occurs. The statistical disparity may discover an underrepresentation of students, younger people between the ages of 18-34, women, blue collar workers or the like.

Next, the statistician can be of invaluable aid to the Court in determining whether or not the underrepresentative class is a cognizable group. In this respect particular statistical aid can be given on the identity issue. Statistical proof regarding the existence of the group as being identifiable by some demographic characteristic is helpful. Social psychology testimony is normally helpful to lay a solid foundation regarding characteristics, or common sets of attitudes. Statistically, it is not essential to prove that everyone in the class has the same attitude but on the average their attitudes should be shown to be different from other cognizable groups and that they have an interest in society which is different from other cognizable groups in society. The statistician can be extremely helpful in this regard. Impressive studies and literature[39] in social science and statistical model demonstrating discrimination and cognizability of educat-

ion, age, race, sex, occupation and income, was utilized in Johnson v. Durante, U.S. District Court, E.D. New York, Docket No. 73C1159, It emphasized the cognizability of the class of persons aged 18-30 years. Also see related work of McConahay, John B., Experimental Design in Political Science, In D. Leege (Ed.) Standards for Design and Measurement in Political Science, New York: Academic Press, 1976, The Uses of Social Science in Trials with Political and Racial Overtones: The Case of Joan Little, Law and Contemporary Problems, 1977; on education and student issues see Yankelovitch, Daniel, The New Morality, A Profile of American Youth in the Seventies (New York: McGraw-Hill), Yamamoto, Kaoru, The College Student and His Culture: An Analysis (Boston, Houghton 1968). Also see numerous articles regarding uniqueness of attitudes of students and/or young people, too numerous to cite.

Lastly, depending on style of the presentation, the statistician can testify relative to the significance of the discrepancy found. Based on the data collected he can either testify that the differences found in any one or many of the various cognizable classes investigated were or were not "statistically significant". This, again, is not a usurpation of the judicial fact finding function. The conclusion of whether the result was the product of chance or discrimination requires the determination of the "significance level". At this point the Court must decide at what point the deviation was greater than would be by normal fluctuation; essentially this refers to the setting of the significance level. It is essential that the statistician make clear to the Court what a significance level is and those that are set by statisticians. In such cases the significance level of .05 is normally set statistically. It is important that the Court be aware and the communication from the statistician is effective to explain simply that if an event has only five chances in one hundred or fewer of occurring it is deemed "significant". At this stage a prima facia case has been established. The statistician in some instances may be able to go the next step, i.e. that the results can be tied to a specific cause, such as lack of random selection, outdated and inaccurate voter registration lists being used as the prime source or a personal selection process involving non uniform standards of exclusion. In any event, it is the statistician's function to make the Court aware that once there is finding of a significance level the "eye brow goes up".[40] At this point a prima facia case has been proven.

The end product of such a contribution by the expert statistician is the establishment of a more uniform and reasonable standard in jury selection and composition. However, regardless of how well prepared and statistically accurateacontribution is it must be effectively communicated to qualified and open ears. Unfortunately, this is not always the case. This demonstration of utilization of statistical testimony and work product in jury selection is only one small area of a continuing unfolding horizon of opportunities to use the talent of the statistician in the Court.

D. A Model Case: Crossroads of Law, Science and Technology

As the days go by more and more of an awakening occurs with regard to the crossroad which the Courtroom provides for the intersecting of law, science and technology. In a contemporary society that continues to speculate about the legal, scientific and technological problems of outer space?[41] the reality of the presence of science and technology in law is no better appreciated than a visit to a Courtroom in 1978. Our present society has become so sophisticated with the advances of the worlds of science and technology that there is a tendency for the weaknesses to surface in a Courtroom rather than in a laboratory. This presence now surfaces both in administrative proceedings as well as in the Courtroom itself.[42] The difference between scientific and technical disagreement and challenge in the laboratory as compared to that which occurs in the Courtroom is the additional factor of the law.[43] The increasing presence of science and technology in the Courtroom, whether it be to resolve a dispute left unsettled in the laboratory or whether it be in response to an invitation from the law to decide a legal principal,clearly demonstrates the need for development of uniform standards which all participants may be sufficiently schooled in and able to communicate.

A fine example of the confusion, conflict and principals discussed can be simply demonstrated by the use of a legal model: New Jersey Sports and Exposition Authority v. The Borough of East Rutherford, Superior Court of New Jersey, Law Division; consolidated with the City of Newark, et als. v. Natural Resource Council, et als., Superior Court of New Jersey, Appellate Division, Docket No. A-3311-72. Here the epitome of the cross roads of science and technology in a courtroom is demonstrated. Legal

issue was simple: the location of the mean high water line on a piece of property adjoining a watercourse.[44] Although the law is simple and clear, the standards and application of the proofs is a tidy mess.[45] The law called upon major scientific and technological fields, including but not limited to surveying and mapping, photogrammetry, hydrology, engineering, biology, botany, geology, computer modeling and statistics. Remote Sensing made its pitch.[46] Hydraulic modeling of the river basin was presented. The end result of the cumulative testimony was extremely diverse. The experts for the State of New Jersey contended that through their methodology and technique 90 percent of 25 acres of the property in question could be deemed to be below the mean high water line and therefore in sovereign ownership. On the other hand, the team of scientists and technicians for the record owner emphatically declared that through their methodology 90 percent of the same property was above the mean high water line. The difference--not the law--science and technology! The State, conducting an experiment, utilized aerial infra-red photography and interpretation in an attempt to establish a botanical mean high water line. Other Courts have found this approach "intellectually fascinating" however, scientifically unacceptable and legally rejected.[47] An attempt to set up signature tones related to plants figure was made; a try correlate the same to relative title innundation was presented. A barrage of reputable scientists criticized the methodology as being insufficient and unreliable scientifically. A representative of the National Ocean Survey testified as to their lack of success with vegetation approaches in establishing mean high water lines. On the other hand, an affirmative approach of delineating the line was put forth by the property owner utilizing conventional surveying techniques with various up to date refinements and verification from other fields of expertise including the ecological, aerial photography and interpretation, hydrology and computer modeling. The phenomena of the tide and its intersection of the land was presented in detailed scientific and technical manner by a team of related experts. A concerted--attack was incurred relative to the tolerances in tide gauges, surveying equipment and varying tidal epochs.

In the midst of this atmosphere a qualified and reputable statistician, John O. Rawlings Ph.D. was called on to do a statistical analysis of both approaches utilized, i.e. botanical v. conventional surveying. Dr. Rawlings educated the Court relative to the to the various acceptable statistical approaches used to test a scientific hypothesis: a special emphasis was given to botanical statistical analysis. He then proceeded, using graphs and charts, to demonstrate a twofold presentation: disproving the accuracy of the State's botanical line and substantiating the municipalities line. A strong foundation was laid. The opposing procedures used by the parties in gathering the relevant status was reviewed. The State's methodology was attached from a statistical point of view: the alleged correlation of elevation and signature was based on incorrect and insufficient elevation data; it was based on no quantitative measurements. Rawlings concluded that the probability of a green phragmities being above mean high water, at an elevation of 3.1 was between 1 and 90 percent (pink was between 16 percent and 53 percent and red between 37 and 100 percent). He stated that they were unacceptable scientifically: these "confidence intervals are very broad and not at all that definitive."[48] There was insufficient data to lead to the conclusion "that elevation and signature are related."[49] The report was also fatally defective relative to it's attempt to transfer data from a southern to a northern New Jersey marsh. Finally, it was concluded that the statistical data presented "in this report * * * is not sufficient evidence to indicate that phragmities signature types are valid indicators of relative signature innundation or such a relationship exists."[50]

The State attempted to show in a non-statistical manner comparison between it's botanical line and a mean high water line drawn by the N.O.S. in other areas of the river basin. The expert's statistical analysis proved no consistency between the two lines and no inference that could be drawn.[51] Certainly there was no transferability of the attempted comparison study to the property in question. (XII: 37-6-6). This expert fortified his conclusion by three different kinds of analysis and exhibits for demonstrative purposes.

On the other hand, statistics prove to be an effective tool in verifying the accuracy of the line delineated by the coventional technique. Dr. Rawlings first analyzed the precision of the spot elevations taken by the surveyor on the property. The surveyor had previously testified that they were accurate to .05 feet. The Rawling's analysis found the probable error to be .051. The expert also studied the distribution of elevations relative to the Borough's mean high water line and the State's phragmities

green line from the stand point of cumulative frequency of the elevations, both landward and seaward, of the respective lines. The conventionally drawn line was far superior. Eighty-nine percent of the surveyed points which were seaward of the Borough's line were less than 3.2, the mean high water line elevation. A distinct pattern was shown of a shortage of elevations less than 3.2 feet in the first 50 feet landward of the line. Field observations proved the existence of a berm in this area.

More and more statistical analysis is brought to life in the Courtroom by the expert. Although deep, complicated and, to a great extent, beyond the qualification of the lawyers and Court, the testimony can be ably communicated and clearly demonstrated by exhibits. This testimony proves to be an extremely valuable tool in clarifying the differences and the reliability of one method as against the other. No usurpation of the Court's function occurs.

III. Guidelines for the Statistical--Computer Expert.

A. General Overview.

Before journeying from preparation to the witness stand, it is important that the statistical--computer expert witness have a thorough overview of his position in the Courtroom. In this respect the areas discussed relative to the subject at issue. being a matter of expertise, as set forth in Section I, should--------be thoroughly reflected upon. It is also important that the expert review and reflect upon the differences between expert and lay witnesses relative to opinion testimony, together with the sections dealing with qualification and presentation of expert testimony. Next, it is important that the discussion regarding the laying of proper foundations to preparation for statistical-computer evidence, as discussed in Section II B be studied.

Thus, the sequence is subject matter, expert and foundation. Preliminarily, determination must be made whether or not the case is a proper one for expert testimony and secondly that the proposedexpert is properly qualified. Then the foundational considerations can be initially analyzed.

B. Preparation

1. Preliminary Stage

a. Know the Problem

As a conservative estimate, preparation is three-fourths of the challenge in presenting effective and persuasive statistical-computer evidence. Contrary to Courtroom mythology, there are few surprises for the well prepared adversary team. Preparation is the prime means to the communication of the facts and opinions. It is the path to the truth.

At the preliminary stage of preparation, the witness and attorney must become thoroughly acquainted with the problem at issue. Too often this basic principle is overlooked. Everyone concerned should be informed as to the general and specific factual and legal issues at question. All should be acquainted with the technical, scientific or specialized field of knowledge required by way of probative expert evidence. This very basic familiarization is the first stone in the foundation.

b. Attorney-Witness Communication

Too often there is little communication between the potential witness and the attorney at the preliminary stages. This is the period of basic understanding and communication between the two. As both will constitute a team ultimately before the administrative or judicial body, it is crucial that frank, truthful and unrestricted communication channels be set up initially. As time passes the two should establish a well tuned and harmonized instrument. It will be necessary for both to educate the other in their respective fields (i.e. Law, statistical or computer science). As a result the lawyer should become a "quasi expert" and the expert a "quasi attorney". But remember: neither should ever attempt to usurp the true expertise of the other or embark on an ego trip.

c. Know your Expertise

The scope of the expertise of the witness in question is basic but too often abused. Both the attorney and the witness should be well acquainted with the witness's qualifications from the outset. This will aid in avoiding the overflow into areas beyond the scope of the expertise in question. Ultimately, before an expert will be permitted to testify regarding opinions

of any nature, the trier must be satisfied that the witness is competent and qualified to testify on the subject matter in question. Thus, at a preliminary stage, the following basic areas of qualification must be established: (1) academics - education and training; (2) professional and technical experience; (3) present position and function; (4) recognition of expertise by other administrative, judicial or legislative bodies; (5) teaching or governmental positions; (6) publications, books or articles written on the particular subject; (7) licenses or registrations held; (8) memberships held in professional societies; (9) accomplishments.

After the basic qualification the expert should prepare for more detailed questions concerning the type of work done in his profession. This will help to establish specific qualifications in the area at issue. As part of this foundational preparation the expert must weave into the qualifications expressed the particular familiarity with the essential elements of the facts in question as they relate to the expertise involved.

d. Know The Site Specific Facts

When the subject matter involves a particular site or location, failure to visit that site is an invitation to instant destruction. A complete familiarization with the site established from the first hand knowledge of visitation, inspection and investigation thereof is a prime value. An acquaintance with every aspect of the location is never too large an order. Likewise, when the expertise will relate to a subject matter the same type of familiarization with that subject matter, instrumentality or object is important.

2. Specific Studies and Reports

The middle stage of preparation is expended in the time consumming studying, analyzing and formulating conclusions relative to the field of expertise in that particular case. This calls for hours of field, laboratory or computer time. Particularly relevant here is the discussion appearing in Section II B supra, regarding the five foundational stages in cases involving computer type evidence (generation and assembly, input, operations, storage and retrieval and output). Detailed and thorough records should be kept regarding time involved, work done, presence of others and certainly dates and times. If photographs are used, they should be labeled by date, time and direction of the view. A specific record of the equipment and type of film utilized is advisable.

The final report should be preceded by preliminary reports from the expert and input from the attorney and other collateral expert witnesses working on the case.

3. Final Lap

a. Preparation of Questions and Answers The final stretch of preparation will occur at the most immediate point prior to the hearing. At this stage most of the polishing is done. Of prime concern is the exchange of potential questions and answers between the attorney and witness. Absolutely no question and, more importantly, no answer should be a surprise to the other at the ultimate hearing. Each and every question should be carefully framed; each answer should be expressed particularly. Revision, alteration and reversal is permissible now for the last time.

b. Cross-Examination Prep The preparation for cross-examination should consist of an attempt to assimilate what can be anticipated under adversary cicrumstances. Potential areas of weakness, uncertainty or confusion must be pinned down now. The approach or type of answers to be given must be reviewed. It is beneficial for the attorney and witness to prepare by way of drilling of actual cross-examination questions and answers. This enactment, with its attendant pressure, will help both to handle the crucial moments which will follow in the Court or hearing room. However, bear in mind that the witnesses testimony should sound natural and unrehearsed. Do not memorize testimony!

c. Demonstrative Evidence Preparation of demonstrative evidence lends visual appeal and makes verbal testimony more vivid. Such visual aids are extremely effective, particularly when used to support the testimony of the statistician or computer scientist. Such evidence usually must be accompanied by oral testimony, since visual aids normally are not admissible into evidence in and of itself. Accordingly, the preparation stage with regard to such evidence becomes very crucial. Eventually, demonstrative evidence must be authenticated by the witness who testifies to the facts, showing the relevancy of the aid to the case.

There is a distinction between original, prepared and selected materials. Ordinarily, inspection of the object itself is the most persuasive means of presentation. This is not practical when dealing with computers. Print-outs, programs, grafts, charts, diagrams, models, photo's and maps fall within the prepared category. Much discretion is vested in the judge relative to their admissibility. Therefore, establishing the relevancy of the aids in the preparation stage becomes all the more important.

Photographs may be identified by a witness in order to portray the facts at a given point in time. Although the witness need not be a photographer nor know anything about the time and conditions of the taking, it is usually advisable for the witness to be familiar with technical facts so that a strong foundation may be laid. The photograph must be an accurate depiction of the scene or object. Verification of the photograph is an essential part of the preparation process. Much creative ingenuity can be utilized in this aspect of the preparation. Moving pictures should be utilized only with caution. A greater degree of preparation is necessary to eliminate the possibility of distortion and falsification due to light, angle, speed and position of the camera. A more important strategic consideration is the time it may take and distraction it may cause to set up the projector and screen in the court room.

Experiments may also be conducted, however, they should be cautiously considered. The factors of confusion, delay and possible failure weigh high in the arsenal of considerations. On the other hand, the persuasive possibilities of this type of evidence present a challenging opporutnity in the statistical-computer fields.

In some instances an actual view of the equipment, object or site by the court may be the best approach. This can only be done when the operation, appearance or condition of it is relevant to the issues involved. Such an approach is usually disruptive of the pace and movement of the trial or hearing and must be weighed accordingly. Consider substituting or supplementing such a view for or with photographs, maps, diagrams and charts.

C. In Court - On The Stand

1. Be Yourself

The most effective witness is a relaxed one. He appears as a genuine individual. The best advice in this regard is "be yourself." Avoid excessive egotism or modesty. Let your expertise, familiarity with the issue and opinions shine through.

Normally conservative dress is recommended for the expert in demanding respect. However, each particular situation may vary in this regard.

Nervous habits should be avoided. For example, chain smoking, in or out of the hearing room, make's the expert appear uncertain and unsettled.

2. Communicate

The major pitfall for the statistical or computer expert is that the testimony given may be above and beyond the grasp of the trier of fact. Communication, even to the extent of "coming down" to the level of the trier of fact, is important. The expert witness is usually the best judge of whether or not communication is occurring and if not, how to correct it. Many subjective factors are involved.

3. Don't Volunteer

The expert should attempt to answer all questions, on direct and cross-examination concisely. Long-winded, complicated answers should be avoided. Long details before answering should be avoided. Cross-examination should be limited to "yes" and "no" answers if possible. When such a reply cannot be given the expert should say so and explain why as succinctly as possible.

Under no circumstances should the witness volunteer or blabber on. This could open up unexpected troublesomeareas for the follow up of the cross examiner.

4. Don't be Adversary

The expert should be even tempered during the testimony. Antagonistic, hostile or adversary responses are taboo. The witness should not attempt to display a quasi knowledge of the law.

5. Be Truthful

The key determinant is the expert's ability to convince the finder of fact of the

truthfulness of his conclusions and opinions. Thus, all of the pitfalls which will discredit that goal should be avoided. Lack of knowledge or ability to express facts or opinions in unanticipated areas should be frankly admitted. Bluffing or a playful avoidance of such admissions can lead to "project downfall". Credibility diminishes! Don't try to bull-throw!

If the attorney and witness have prepared well, their ability to convey truthful facts from the stand will be totally enhanced.

Lastly, take a deep breath--say a prayer! See you in Court!

IV. Reasonable Probability: Total Insight

An evaluation of the past and present of the lawyer and statistical-computer expert in Court is appropriate for commitment to the future. To attempt to do this in the phraseology of the law or a probability theory of the expert would be futile at this point. A Jesuit theologian, Bernard J. F. Lonergan, S.J., provides a pivotal method for the process. He suggests a dynamic continuing process from point of inquiry to insight. This method emanates from the pinnacle of man's understanding-self-appropriation. The four essential stages of this consciousness are experience, understanding, judging and deciding.[52] Within these four dimensions, insight through self-appropriation can be obtained relative to the relationship between the lawyer and the statistical-computer expert. The result will be a total insight which will lead to a new horizon, i.e. truth in the Courtroom and ultimately justice.

The starting point to obtaining an effective interface is the mutual experience of each. A mutual consciousness in a phenomenological sense exists. The event of experience has occurred, at least for the participants in the computer-science and statistics Symposium on the Interface. With an exertion of this conscious self, the research, history and interpretations set forth in the foregoing sections provide a functional path in the understanding stage. The contemporary search for truth and persuasion in adversary legal proceedings necessitates a mutual understanding between the lawyer and the computer scientist and statistician. The foregoing contributions from the lawyers standpoint will hopefully be an instrument in obtaining the mutual goal. Likewise, past and present contributions being experienced personally and by the legal profession from the computer statistical experts provides a strong aid of understanding in the relationship. Through this method a phenomena beyond mere formality will occur. The processes of experience and search for understanding of the past and present provide the backdrop for the ultimate dimensions of judgment and decision.

The very creation of the field of Jurimetrics is indicative that this process has been afoot for some time. The experiences, understanding and judgments of others have already made a commitment to the basic relationship. However, today we are requestioning and moving to a new horizon, i.e. the melding of the evidential rules and methods of the law with the science and technology of measurement. This is the challenge. Experience and understanding to this point has surfaced the need for the various professions and advocations involved to understand basic concepts of the other. This presentation from a legal prospective to the scientist and technician justifies certain judgments that can be made now. There is a need to increase the qualifications of all involved--the lawyer, expert and the Court. There is a necessity to achieve a higher degree of communication in the relationship. It is essential that some uniform standards be set.

The past and present demonstrates that the law schools throughout the country are dedicating more and more courses and seminars dealing with jurimetric topics. Generally, they are "computer-computerized information systems, computers and the law, law and biology, technology assessment, communications law, law and technology, computerized legal research, law and medicine, legal process and technological change, sociology of law, and environmental law. * * * It's already happening."[53] An evaluation process is now occurring: " From the legal prospective, the scientist contribution can be seen as primarily one of establishing facts: the ones necessary to generate public policy (legislative facts), and the ones necessary to implement policy (adjudicative facts)" * * * what is needed is an examination of all courses to determine whether they are helpful in preparing tomorrow's lawyer to marshall facts and to grasp technical vocabulary in context. * * * only with exposure to the commonalities and differences of various disciplines can tomorrow's lawyers be prepared to meet the growing number of

diverse problems which demand solution."[54] A commitment should be made to incorporate explicit treatment of scientific courses, such as Statistics for Lawyers.

Ignorance must be pleaded relative to an understanding of what legal related subject matters are being taught to the computer scientist or statistician. This symposium regarding the interface is a first experience with the experts legally related educational programs. It has stimulated a new inquiry which will strive for insight.

The criticism of the Courts qualifications in the past and present are quite uniform indeed. Judges, like the attorneys and experts, are only human beings. They have come from limited number of backgrounds normally, ie. political, prosectutional, governmental or general practice. Additionally, it must be recognized that the judicial process involves many subjective, non legal or scientific, factors and motives. Unfortunately, justice is not always the ultimate result. Thus, experiencing and understanding the base of qualification inherent in the Court system some judgments can be made. The proposals to date are limited and basically have not been implemented. Some have suggested for special Courts where judges familiar with computer technology can dispense justice when the computer is involved or its generated product being submitted into evidence.[55] For many years qualified evidence treatises have called for the utilization of Court appointed experts and scientificly qualified jurors or fact finders.[56] Mandatory continuing education for lawyers, computer scientist and statistician and judges, alike, should be considered. This would appear to be a very meaningful technique of increasing competency.

The increased level of qualification will enhance the channels of communications. Communication must be on an in and out of Court basis. The foregoing article has demonstrated the breakdown in communication that presently exists between the attorney and expert and more oftenly the expert and the Court. Certainly, the out of Court communication is even much less. The lack of qualification factor has been detrimental to a movement from the experience to a stage of understanding. The events of experience of this sad situation are becoming more and more prevalent. A true sense of understanding is starting to develop legally and scientifically.

Communication must not be just a vogue word[57] nor a dictionary of legal, scientific and technical terms.[58] It is a process. A process which is essential for the interface. The acts and instances of transmitting information, whether verbal or written, must be fluid, continual and cumulative. The gravamen of process thought is a method of "becoming" and not "being".[59] "The very act of becoming is what is real".[60] From such a process a set of symbols, signs and behavior will result. This will develop into--more uniform standards for the interface.

The burden of effectuating such a process is slow and often seemingly without reward. "As for you, I lay up to your credit a rich reward as though yourselves had accomplished it". Recall Rabban Gamaliel speaking to those who bear the burden of communal work, to wit: "You who bear this burden, if your labors you have tried once and despite your efforts have accomplished nothing, do not throw up your hands in despair * * *".[61] Understand, make your judgment and give your commitment to communication.

From qualification and communication uniform standards will start to develop. It may be simultaneous or it may trail conservatively behind. It will differ relative to various aspects of the interface. Standards will not be legal nor will they be scientific or technological. They will be a combination of this interface. A degree of jurimetric uniformity in the field of reception scientific and technical evidence is the prime aim. A degree of uniformity was the first result for the foundational matters, i.e. the acceptability of the subject as one of expertise, the qualification of the expert and the basic collection and processing of data. A degree of uniformity has been and will continue to grow in the area of the expression and utilization of factual and opinion evidence in the computer science and statistical fields.

Utilizing Lonergan's methodology one will start and end with an inquiry. Why? Why the general rejection of probability theory to problems of legal relevance? Why did the Court in Collins reject mathematical evidence on the issue of identification? Why should students be rejected as a cognizable groups when the undisputed scientific and technical evidence is contrary? The answers lie in the process. The experience, understanding and judgments expressed are not intended to

impose decision on the audience. Decision, being commitment, must be on a more personal basis before it becomes communal. Hopefully, this contribution to the law, science and technical interface will aid. No one solution or formula is recommended. A direction for the process is attempted to be imposed, i.e. qualification, communication and stress toward uniformity of standards. The scope of the decision is wide. The direction in all reasonable probability, to borrow a word from each segment of the interface, will result in total insight.

Footnotes

1. McCormick, Charles T., Law of Evidence, Ch. 3, Sec. 10-18, P. 19 et seq; CF Rule 701, Federal Rules of Evidence.

2. McCormick, supra, Sec. 13.

3. 7 Wigmore, Evidence, Sec. 1923 (3d. Ed. 1940).

4. 11 Moore's, Federal Practice, Sec. 702 10 (1) (1976)

5. Lempert, Richard O. and Saltzburg, Steven A., A Modern Approach to Evidence, Ch. 3, P. 140, et seq. (1977).

6. Lempert, supra, P. 998.

7. McCormick, supra, P. 363.

8. McCormick, supra, P. 363

9. Rule 702, supra.

10. Lempert, supra, P. 932-933.

11. Allen, Laymane, Preface, Jurimetrics Journal, P. 72, A.B.A (Winter 1975).

12. Arthur, R obert S., The Computer and The Practice of Law: Litigation Support, Vol. 63, American Bar Association Journal, P. 1737. (Dec. 1977).

13. Arthur, Supra, P. 1739 et seq.; also see Computers Are Gaining on the Legal Pad as Lawyers aid in Complex Court Cases, Wall Street Journal, P. 13, Nov. 2, 1977.

14. Gerjuoy, Edward, The Relevance of Probability Theory to Problems of Relevance, Jurimetrics Journal, A.B.A., Fall, 1977).

15. McCormick, supra, Sec. 171, P. 367.

16. Id., also see Fairley and Mosteller, A Conversation About Collins, 41 U. Chi. L. Rev. 242 (1974).

17. Richardson, supra, P. 131.

18. Gerjuoy, supra.

19. Ginger, Due Process in Practise or Whatever's Fair, 25 Hastings L. J., 897 (1974).

20. Hurtado v. California, 110 U.S. 516, 4 S. Ct. 111, 28 L. Ed. 232 (1884); Alexander v. Louisiana, 405 U.S. 625, 92 S. Ct. 1221, 31 F. Ed. 2d. 536 (1972); Smith v. Yeager, 465 F. 2d 272 (3rd Cir. 1972).

21. Taylor v. Louisiana, 419 U.S. 522, 95 S. Ct. 692, 42 L. Ed. 2d 690 (1975); Peters v. Kiff, 407 U.S. 493, 92 S. Ct. 2163, 33 L. Ed. 2d 83 (1972); Neil v. Delaware 103 U.S. 370, 26 L. Ed. 2d 567 (1881).

22. 28 U.S. C.A. 1861.

23. State v. Rochester 54 NJ 85 (1969)

24. Thiel v. Southern Pacific Co. 328 U.S. 217, 66 S. Ct. 984, 90 L. Ed. 1181 (1946).

25. Arnold v. North Carolina, 376 U.S. 773 (1964); Hernandez v. Texas 347 U.S. 475, 74 S. Ct. 667, 97 L. Ed. 336 (1954).

26. Taylor v. Louisiana, supra.

27. Glaser v. United States, 315 U.S. 60 (1942); Dow v. Carnegie-Illinois Steel Corp., 224 Fed. 2d 414 (3rd Cir. 1955, Cert Den. 350 U.S. 971 (1956); Crawford v. Bounds 395 Fed. 2d 297 (4th Cir. 1968), Carmical v. Craven 457 Fed. 2d 582 (1971).

28. Carter v. Jury Commission of Green County, 396 U.S. 320, 90 S. Ct. 518, 24 L. Ed. 2d 549 (1970).

29. Taylor v. Louisiana, supra.

30. Thiel v. Southern Pacific Co., supra.

31. E. G. Government of Canal Zone v. Scott 502 Fed. 2d 566 (5th Cir. 1974) (Military Personnel); Rawlins v. Georgia, 301 U.S. 899 (1906) (Lawyers, Doctors and Firemen).

32. People v. Marr, 67 Misc. 2d. 113, 324 NYS 2d 608 (1971); People v. Attica Bros. 79 Misc. 2d. 492, 359 NYS 699 (1974); Paciona v. Marshall, 45 App. Div. 2d. 462, 359, NYS 2d 360 (1974) Aff'd 35 NYS 2d 289 (1974); Anderson v. Casseles, 531 Fed. 2d. 682 (2d Cir. 1976).

33. Brown v. State, 58 Wisc. 2d. 158, 205 N.W. 2d 566 (1973).

34. Cf. State v. Carter, Superior Court of New Jersey, Law Division, #167-66 (Oct. 12-13, 1976) and State v. Porro, Superior Court of New Jersey, Law Division, Docket No. S-1320-75 (July 20, 1977).

35. Himelerick, Richard G., Federal Courts-Juror Selection-Underrepresentation of Young Adults on Juror Source Lists, 19 Wayne L. Rev. 1287, 1296 (1973); United States v. Gargan, 314 Fed. Sup. 414 (W.D. Wis. 1970) aff'd sub. nom. United States v. Gast 457 Fed. 2d 141 (7th Cir. 1970) Cert. Den. 406 U.S. 969 (1972).

36. Kairys , David, Juror Selection: The Law, A Mathematical Method of Analysis, and a Case Study , 10 Am. Crim. L. Rev. 771, 780 (1972).

37. Turner v. Fouche, 396 U.S. 346, 9 S. Ct. 532 24 L. Ed. 2d 567 (1970), Sim v. Georgia, 389 U.S. 404, 88 S. Ct. 523, 19 L. Ed. 2d 634 (1957); Witcher v. Peyton, 405 Fed. 2d 725 (4th Cir. 1969), Coleman v. Alabama, 389 U.S. 22, 88 S. Ct. 2d, 19 L. Ed. 2d 22 (1967).

38. Jospovice, Martin L., and Sperlich, Peter, W., Grand Juries, Grand Jurors and the Constitution, 1 Hastings Const'l L. Q. (Spring 1974) and Sperlich, Peter W. Statistical Decision Theory and The Selection of Grand Jurors: Testing for Discrimination in a Single Panel, 2 Hastings Const'l L. Q. (Winter 1975).

39. Faust, Richard and Carlson, Josephine, The Impact of Age and Other Stratification Variables on Attitudes Toward Justice, Publication #CR32, Center for Responsive Phychology Brooklyn College, CUNY (1977).

40. McConahay, supra.

41. Young, Roland L., The Law In Space-Age Year 21, American Bar Association Journal, P. 1424 (Oct. 1977).

42. See Finkelstein, Michael O., Regression Models in Administrative Proceedings, 86 Harv. L. Rev. 1442 (1973); McKenna, W. M., The Military Defense Counsel's Use of Scientific Evidence and Expert Witnesses 18 Air Force Hel. R. 48 (Fall 1976) and Applichian Power Co. v. Train 545 Fed. 2d 1351, (4th Cir. 1976).

43. Leventhal, Harold, Environmental Decision Making and the Roles of the Courts, 122 U. Penn. L. Rev. 509 (1974).

44. Legally the Mean high water line is the boundary between private and sovereign ownership. Martin v. Waddell's Lessee, 16 Pet. 367, 41 U.S. 346, 10 L. Ed. 997 (1842); O'Neill v. State Highway Dept. of New Jersey, 50 N.J. 307, 235 A. 2d (1967).

45. Porro, The Invisible Boundary--Private and Sovereign Marshland Interest 3 Natural Res. L. 512, (1970) Porro and Teleky, Marshland Title Dilemma: A Tidal Phenomena 3 Seton Hall L. Rev. 323 (1972); Porro, Meadowland Title Dilemma: Two Decades Later, Meadowlands: U.S.A. (1977.)

46. Latin, Howard A. Tannehill, Gary W. and White, Robert E., Remote Sensing Evidence and Environmental Law, 64 Calif. L. Rev. 1359-1364, 1368 and 1370 (1976); Osterhoudt, William L., Remote Identification by Forward-Looking Infrared: A New Scientific Challenge, Legal Notes and Viewpoints Practicing Law Institute, July 15, 1977.

47. 289 Dolphin Lane Association v. Township of South Hampton, New York Court of Appeals, Decided January 2, 1975.

48. See Transcript of Testimony (XXVIII: 85-12-16).

49. Id. (XXVIII: 88-1-16).

50. Id. (XXVIII:109-4-7).

51. Id. (XIII 28-22-25 and XII: 56-13-13)

52. Lonergan, Bernard J. F., S.J., Method in Theology, Herder and Herder, New York (1972) and Insight, a Study of Human Understanding, Philosophical Library, New York, (1957), Liddy, Richard M., Lonergan's Method, American P. 68, Aug 5, 1972.

53. Issue Dedicated to Jurimetrics in Education, 16 Jurimetrics Journal, P. 71-73 (Winter 1975).

54. Field, Thomas G., Training Lawyer's To Cope with Science, The Profession Can no Longer Afford to Hide From What it Doesn't Know, Learning and The Law, Section of Legal Decuation and Admissions, American Bar Association P. 31 (Fall1977).

55. Ewing, David, Is It Time for a Computer Court? The Robotts are Coming! Juris Doctor, P. 18 (Dec. 1977).

56. McCormick, Charles T., Evidence, supra P. 34-38.

57. Morris, William and Mary, Harper Dictionary of Contemporary Usage, P. 624, 625 (Harper 1975).

58. Dictionary of Scientific and Technical Terms (McGraw-Hill 1975).

59. Whitehead, Alfred North, Science and the Modern World, McMillan New York, 1925 also see Modes of Thought, Free Press, New York, 1938 and Process and Reality, McMillan, New York, 1929.

60. Lee, Bernard, S.M., The Becoming of the Church, A Process Theology of Structures of Christian Experience, Paulist Press, New York, 1974, P. 54.

61. Goldin, Judah, The Living Talmud, The Wisdom of the Fathers, P. 83 (Mentor, 1957).

BIOGRAPHY

Alfred A. Porro, Jr. is an attorney specializing in trial of technical and scientific cases in the field of public law. He has been trial counsel in extensive tideland and marshland litigation, public authorities rate schedule cases, environmental hearings and constitutional attacks based upon expert supportative testimony.

His office is located at 10 Stuyvesant Avenue, Lyndhurst, New Jersey 07071. He is a graduate of Rutgers Law School and a member of the American Bar Association Science and Technology Section's Committee on Legal Reception of Scientific Evidence. He serves on the Board of Editors of the New Jersey Bar Journal of the New Jersey State Bar Association. He is also former special counsel to the United States Commission on Marine Science, Engineering and Technology.

He has written extensively on his field of expertise and has lectured at various law schools and universities throughout the United States and abroad.

WORKSHOP 6

COMPUTING FOR BAYESIAN METHODS

Chair: Joseph B. Kadane, Carnegie-Mellon University

APPROXIMATING MARGINALS FROM NON ANALYTICALLY INTEGRABLE JOINT DENSITIES

by

MARCEL G. DAGENAIS ET CONG LIEM TRAN

Université de Montréal

ABSTRACT

This paper discusses a method to approximate marginal univariate densities, when these cannot be obtained analytically from the associated multivariate density. This is a problem often encountered in Bayesian statistics. The technique is particularly useful when dealing with joint densities of more than three dimensions, since numerical integration with a minimum of accuracy then becomes very costly. The first part of the paper summarizes briefly the technique and its rationale. The second part mentions several problems of numerical analysis raised by the approach.

Problems of inference in Bayesian statistics often lead to multivariate joint densities that cannot be integrated analytically to obtain the marginals of the individual parameters. For cases involving more than five or six dimensions, integration by numerical techniques is, as far as we know, virtually impossible, both for reasons of costs and accuracy. This computational problem is no doubt, presently, one of the major drawbacks for the diffusion of Bayesian procedures in applied statistical work. This is an area where the contributions of numerical analysts could be most helpful to statisticians, econometricians and other users of Bayesian techniques.

This paper discusses a method that has been used with some success to approximate the marginal densities of individual parameters from "well-behaved" non integrable joint posteriors containing up to six parameters. The procedure is generalizable to even greater numbers of dimensions. The first part of the paper describes briefly the approach[1] and the second part lists some of the numerical problems that may arise from the use of this method. Our suggested technique must, by no means, be seen as a definitive solution to the problem at hand. We consider it, on the contrary, as a first effort that might encourage further research leading to more significant contributions on this very important problem.

I. The Method

The fundamental idea underlying the approach is, in order to obtain the marginal densities of the parameters of interest from the non integrable joint density, to perform appropriate changes of variables such that the

[1]. A more extensive description of this method, with numerical and graphical examples, can be found in Dagenais and Tran (1977).

shape of the joint density of the new variables is as close as possible to the shape of the multivariate normal density. If the transformed density is very close to the normal, the marginals of the transformed variables can then be closely approximated by standard orthogonalization proceduresused to derive univariate marginals from a multivariate normal. Yet, the changes of variables performed must also be such that it remains possible to make, in reverse, additional univariate changes of variables on the marginals of the transformed variables so as to obtain the marginals of the parameters of interest. Our procedure entails five steps. To make matters more easily intelligible, let us assume that the joint density under consideration [$p(\beta,\Sigma)$] involves a k-dimensional vector of parameters (β) appearing in the means of some stochastic variables and an (m x m) covariance matrix (Σ) .

1. The first step is not systematic. It will vary according to the density under consideration. It consists of making a first set of ad hoc transformations to enhance the resemblance of the joint density to the multivariate normal.

   One useful type of preliminary transformation that can be applied to the elements of the Σ matrix, when these may be considered as nuisance parameters, is discussed in section 1. of Part II.

   The changes of variables performed on the parameters of interest should however affect only one parameter at a time[2].

2. The second step consists in fitting a multivariate normal density to the transformed density [$p(\beta^*,\Sigma^*)$] by finding its mode and evaluating its matrix of second derivatives. If the transformed joint density were perfectly normal, it would be easy to obtain the exact marginals of each of the transformed variables [i.e. $p(\beta_i^*)$, $(i=1,\ldots,k)$ and $p(\sigma^*_{j,\ell})$ $(j,\ell=1,\ldots,m)$] by applying standard orthogonalization procedures. Since, however, the joint density obtained at the end of step 1 is only approximately normal, only a first approximation to each marginal may be obtained by taking the conditional of each transformed variable, given the modal values of the other orthogonalized variables [e.g. $p(\beta_1^*|\beta_2^o,\ldots,\beta_k^o,\ \sigma_{11}^o,\ldots,\sigma_{mm}^o)$] where the β^os and σ^os are the modes of the other orthogonalized variables.

3. Measures of skewness and kurtosis are then computed for each approximate marginal density obtained at the end of step 2. Then, further changes of variables are made on each transformed variable (β_i^* $(i=1,\ldots,k)$ and $\sigma^*_{j,\ell}$ $(j,\ell=1,\ldots,m)$ using Johnson's (1949) system of transformations. These transformations are such that when applied to the approximate marginals obtained at the end of step 2, the resulting densities have all first four central moments identical to those of the standardized univariate normal distribution.

   These same transformations are then applied to $p(\beta^*,\Sigma^*)$ found at the end of step 1, to obtain $p(\bar{\beta},\bar{\Sigma})$. Note that for all parameters of interest, there is still a one to one correspondence between the relevant elements of $(\bar{\beta},\ \bar{\Sigma})$ and those of $(\beta,\ \Sigma)$.

4. A new normal distribution is then fitted to the new joint density $p(\bar{\beta},\ \bar{\Sigma})$ and after applying the same orthogonalization procedure as in step 2, second round approximations of the marginals are again obtained by using the appropriate conditionals.

5. Marginals of the original parameters of interest are then derived from the marginals found at the end of step 4.

The measures of skewness and kurtosis of the marginals obtained at the end of step 4 have been found to yield very satisfactory indices

---

[2]. Or small group of parameters. See below, part II, paragraph 3.

of the accuracy of the approximating procedure. If the suggested approach is valid, the values of these indices ought to coincide with those of the normal distribution for all marginal densities: discrepancies are therefore good measures of the inaccuracy of our system of approximation.

Applications of the proposed technique [see Dagenais and Tran (1977)] were made to a three-dimensional probit model and to a six-dimensional regression model with incomplete observations[3]. In all cases, the marginals obtained by our method were extremely close to the "true" marginals evaluated by numerical integration[4], while more standard techniques such as the usual large sample approximations to the marginal densities of the parameters would in some cases yield totally unsatisfactory results. Our method took four times less computer time than the 9^5-point Gaussian quadrature routine [Stroud (1971)] used for numerical integration.

II. Problems of numerical analysis

The application of the approach suggested in part I raises several problems of numerical analysis which deserve further study and could be of interest to numerical analysts.

1. The first problem concerns step 1 and consists in finding simple rules for performing appropriate changes of variables, according to the types of probabilistic models considered.

   For example, it seems that in different types of multivariate linear models, where the elements of the covariance matrix are nuisance parameters with no interest per se, the following type of transformation is appropriate: we use a 3x3 covariance matrix (Σ) as an example, but it can be generalized to more dimensions.

   First, write:

   $$\Sigma = \Lambda R \Lambda$$

   where Λ is a (3x3) diagonal matrix with typical diagonal element $\sqrt{\sigma_{ii}}$ (i=1,2,3) and R is a matrix of pairwise correlation coefficients. Then, changes of variables inspired from Fisher (1921) and from Box and Tiao [(1973). p. 413] are made:

   $$w_{ii} = \ln \sigma_{ii} \qquad (i=1, 2, 3)$$

   $$z_i = \frac{1}{2} \ln \left(\frac{1 + r_{i3}}{1 - r_{i3}}\right) \qquad (i=1, 2)$$

   $$g = \frac{1}{2} \ln \left(\frac{1 + r_{12.3}}{1 - r_{12.3}}\right)$$

   where

   $$r_{12.3} = (r_{12} - r_{13}\, r_{23}) / \sqrt{(1 - r_{13}^2)\,(1 - r_{23}^2)}$$

   and

   $$r_{ij} = \sigma_{ij} / \sqrt{\sigma_{ii}\, \sigma_{jj}} \qquad (i,j=1,2,3)$$

2. Once the first changes of variables have been made, it would be useful to have means of verifying that the joint density obtained at the end of step 1 has a shape that does not depart "too much" from the normal. Presumably, functions with several maxima or with very irregular or twisted shapes could not be treated by the technique discussed in this paper.

3. One of the limitations of our procedure is that, except for nuisance parameters (such as the elements of the covariance matrix, in some cases), the transformations made in steps 1 and 3 can affect only small groups of parameters at a time. When the groups contain more than one parameter, step 5 then requires numerical integrations. For any given group, the order of the numerical integration to be performed equals the number of elements in the group minus one[5].

4. We have used as indices of accuracy measures of skewness and kurtosis computed at the end of step 4. Alternative approaches for evaluating

---

[3]. The probit model is discussed in Theil [(1971, pp. 630-31] and the Bayesian regression model with incomplete observations, in Dagenais (1974).

[4]. For example, the 90% confidence intervals coincided almost exactly. Thos found with our procedure were always slightly shorter: they were 4.5% shorter on the average, with a maximum of 7%.

the accuracy of our approximations could possibly be developed.

5. For the univariate numerical integrations performed at steps 2 and 4, bounds had to be set to the conditional densities of the individual parameters. Since the densities obtained at the end of step 2 are sometimes quite asymmetric, the upper and lower bounds must not be set at equal distances from the mode. Reasonable values for these bounds may be found by trial and error. More systematic procedures could probably be devised.

   Note that the same boundsare used to perform Johnson's transformations in step 3.

6. Finally, other transformation systems could possibly be used when that proposed by Johnson does not appear to give satisfactory results. In the limited number of experiments that we have made however, Johnson's system has always yielded good results.

REFERENCES

Box, George E.P. and George C. Tiao, 1973, Bayesian Inference in Statistical Analysis, Addison-Wesley Publishing Co..

Dagenais, Marcel G. and Tran, Cong Liem, 1977, "Numerical Approximations of Marginals from "Well-Behaved" Joint Posteriors, working paper, département de sciences économiques, Université de Montréal.

Dagenais, Marcel G., 1974, "Multiple Regression Analysis with Incomplete Observations, from a Bayesian Viewpoint", in Studies in Bayesian Econometrics and Statistics, edited by Stephen E. Fienberg and Arnold Zellner.

Fisher, R.A., 1921, "On the "Probable Error" of a Coefficient of Correlation Deduced from a Small Sample", Metron, 1, 3.

Johnson, N.L., 1949, "Systems of Frequency Curves Generated by Methods of Translation" Biometrika, no. 36, December.

Stroud, A.H., 1971, Approximate Calculation of Multiple Integrals, Prentice Hall, Inc., N.J.

Theil, Henri, 1971, Principles of Econometrics, John Wiley and Sons, Inc..

---

[5]. In part I, we have assumed for simplicity that the changes of variables of steps 1 and 3 affected only one parameter of interest at a time. In fact, transformations affecting small non overlapping groups of two or three parameters would still lead to a workable procedure.

MATRIX WEIGHTED AVERAGES: COMPUTATION AND PRESENTATION

Herman B. Leonard

Department of Economics
Society of Fellows

Harvard University

Inexpensive methods for computing matrix weighted averages of the form $b^{**}(p)=(H+pN)^{-1}(Hb+pNb^{*})$ for various values of p are developed and illustrated. The central device of simplification is the joint diagonalization of the weight matrices H and N, which permits their weighted sum to be written as a product of fixed matrices and a variable but easily invertible matrix of the form I+pD. This is further developed and a polynomial expansion for $b^{**}(p)$ is given. This permits the application in the current context of the extensive existing knowledge of polynomial roots problems. The methods are applied to the problem of characterizing the ambiguity in the posterior arising from uncertainty about the prior in a standard Bayesian econometrics problem.

INTRODUCTION

Matrix weighted averages of estimated parameter vectors arise especially frequently in Bayesian econometrics, though they are by no means rare in classical inference. Efficient computation of these averages is essential, because exploration of the inferential context often requires deriving the averages for numerous different weights. This paper collects some recent work on the properties of matrix weighted averages, their computation, and their geometric representation. Many of the results on the properties of such averages were developed in a pair of papers by Chamberlain and Leamer [1,2]. That work has been continued and expanded to include the development of a package of computer routines to carry out estimation of a special class of problems in Bayesian inference [3], and it is this experience that provides the substance of the results discussed here.

By a "matrix weighted average" of two kxl vectors a and b will be meant a simple matrix analog of a weighted average of two scalars. If A and B are two positive (semi) definite kxk matrices, then a matrix weighted average of a and b would be given by

$$c = (A + B)^{-1}(Aa + Bb)$$

In the simple scalar case, with k=1, this reduces to

$$c = \left(\frac{A}{A + B}\right)a + \left(\frac{B}{A + B}\right)b$$

$$= \theta a + (1-\theta)b \quad (\theta = A/A+B)$$

a simple weighted average.

Matrix weighted averages arise quite naturally in a variety of contexts in Bayesian econometrics. The simplest and perhaps

best known example is that of the standard regression model with normally distributed errors and a normal prior. Because it is particularly simple, it will be used as the standard example throughout this discussion. The model problem is defined by y, an nxl vector of observations of the dependent variable; X, an nxk non-stochastic matrix of observations of the independent variables; β, a kxl vector of (unknown) true parameters; and ε, an nxl vector of independently identically normally distributed errors with mean 0 and variance covariance matrix $\sigma^2 I_{nxn}$. It is assumed that

$$y = X\beta + \varepsilon$$

The inferential problem is to make statements about the values β. The standard likelihood function induced by the data alone is

$$L_d(\beta|y,X,\sigma^2) = c_1 \exp[(-1/2\sigma^2)(y-X\beta)'(y-X\beta)]$$

This can be rewritten in terms of the error sum of squares at the "least squares point" as follows:

$$L_d(\beta|y,X,\sigma^2) = c_1 \exp[(-1/2\sigma^2)(ess_b + (b-\beta)'X'X(b-\beta))]$$

where $b = (X'X)^{-1}X'y$

$$ess_b = y'y - y'X(X'X)^{-1}X'y$$

In this formulation it is easy to see that b is also the maximum likelihood estimate of β.

The Bayesian investigator has a normal prior distribution for β, defined by b*, a kxl vector of prior means, and λN, a kxk positive (semi) definite precision matrix for β. The prior distribution is thus given by

$$L_p(\beta) = c_2 \exp[(-1/2)(b^*-\beta)'\lambda N(b^*-\beta)]$$

The two likelihoods combine to yield a posterior distribution,

$$L(\beta|y,X,\sigma^2) = c_3 \exp\{(-1/2)[(b^*-\beta)'\lambda N(b^*-\beta) + (b-\beta)'H^*(b-\beta)]\}$$

where $H^* = X'X/\sigma^2$

The point of highest likelihood for the posterior, which must be the posterior mean since with normal distributions the mean and mode are the same, is the vector b** that minimizes the expression

$$(b^*-\beta)'\lambda N(b^*-\beta) + (b-\beta)'H^*(b-\beta)$$

The first order conditions are given by

$$\lambda N(b^*-\beta) + H^*(b-\beta) = 0$$

or $\lambda N b^* + H^* b = (\lambda N + H^*)\beta$

so that

$$b^{**} = (\lambda N + H^*)^{-1}(\lambda N b^* + H^* b)$$

Thus, the posterior mean in the archetypal Bayesian econometrics problem is simply a matrix weighted average of the prior mean b* and the "data mean" b, the "weights" being given by the corresponding precision matrices, normalized by the sum of the precision matrices.

It should be noted that the value σ^2 has been treated as known in the discussion above. While σ^2 is generally the most precisely estimated parameter in models of this kind, it is generally not in fact known. Moreover, while it may not be unreasonable to expect an investigator to have some feeling about the appropriate prior variance covariance structure embodied by N, it is perhaps safer to allow the value λ not to be specified with certainty. Since σ^2 and λ are involved in the same way in the weighting of b and b*, this suggests rewriting the relation for b** as

I] $b^{**} = (pN + H)^{-1}(pNb^* + Hb)$

where $H = X'X/s^2$

s^2 is the usual estimate of σ^2 $(ess_b/n-k)$

$p = \lambda\sigma^2/s^2$

Conditional on p, the posterior mean b**(p)

is given by I. Generally, the ratio σ^2/s^2 is thought to be sufficiently close to 1 that the primary ambiguity in p is from the unwillingness of the investigator to specify a priori with certainty a prior scale λ. [1] If the investigator were entirely sure of the various components of his prior (sure that they correctly stated his true beliefs), then he would be interested in computing only one posterior b**, and the computational task would be simple. The only remaining ambiguity would be in the ratio σ^2/s^2, and this could be dismissed for most problems. Unfortunately, the world is not always so simple. Investigators are rarely so sure about b*, λ, and N that they only wish to know a single posterior estimate. The general problem, then, is that of helping the investigator to explore his ambiguity in an organized and interpretable way. The questions he asks are of the form, "how much difference does it make in the posterior if ...." This paper is a discussion of ways of computing answers to a subset of such questions that can be computed readily using properties of matrix weighted averages. These include questions about changes in λ (or,equivalently, in p) and in b*, but not in N. Thus, this discussion will develop a set of ways of characterizing the effects on the posterior of changing 1) the uncertainty of beliefs in the prior, and 2) the prior location b*. [2] These tools would permit answers to the question, "what would be the value of the posterior for someone who thought β_1 was twice what I think it is?," and to the question, "what whould be the posterior for someone who believed this prior twice as strongly as I do?"

In the next section, a series of simple and inexpensive computational methods for handling matrix weighted averages will be developed. In the final section the discussion will return to the formulation and resolution of questions like those just posed.

COMPUTATION

This section presents a series of formulae for computing matrix weighted averages. All are based on finding the axes of tangency between the two families of ellipses generated by the "weight" matrices. This is called a "joint diagonalization," and it is considered first. Next, a simple computational procedure based directly on the diagonalization is developed. While this procedure is handy for some purposes, it is useful for other purposes to simplify it to a polynomial expression, and this is presented in the third portion of this section.

a. Joint Diagonalization

Since H and N are both positive (semi) definite, each has strictly nonnegative eigenvalues. [3] Thus, there is a matrix S_1 of full rank such that

$$S_1'HS_1 = D_1 \quad \text{and} \quad S_1'S_1 = I$$

where D_1 is a diagonal matrix with positive elements. Since N is positive (semi) definite (psd), $D_1^{-1/2}S_1'NS_1D_1^{-1/2}$ is psd, and there is a matrix S_2 of full rank such that

$$S_2'D_1^{-1/2}S_1'NS_1D_1^{-1/2}S_2 = D_2$$

and $S_2'S_2 = I$

where D_2 is a diagonal matrix with nonnegative elements. Note that

$$S_2'D_1^{-1/2}(S_1'HS_1)D_1^{-1/2}S_2 =$$

$$S_2'(D_1^{-1/2}D_1D_1^{-1/2})S_2 = S_2'S_2$$

$$= I$$

so that $S = S_1 D_1^{-1/2} S_2$ diagonalizes both N and H:

$$S'HS = I$$

$$S'NS = D_2$$

with S of full rank. It is clear from the relation above that S and $D = D_2$ are solutions to the generalized eigenvalue problem

$$NS = HSD$$

From this dual generalized eigenvalue formulation follows immediately the geometry of the joint diagonalization. The columns of S are the vectors along which lie the joint tangencies of the ellipsoids $x'Hx = c$ and $x'Nx = \underline{c}$. Figure 1 gives an example in two dimensions. To show that the tangencies lie along the lines generated by the columns of S, note that (as is clear from Figure 1) $\lambda_1 s_1$ solves "minimize x'Nx subject to $x'Hx=c_1$" and $\lambda_2 s_2$ solves "max x'Nx s.t. $x'Hx=c_1$." Formulating this problem as a Lagrangean,

$$L(x) = x'Nx - \lambda(x'Hx-c_1)$$

for which the first order conditions are

$$\frac{\partial L}{\partial x} = 2Nx - 2\lambda Hx = 0$$

or $Nx = Hx\lambda$

This is exactly the generalized eigenvalue problem NS=HSD, so that the solution x must be one of the columns of S, and conversely, the columns of S must the directions of tangency between the ellipses generated by N and H.

b. A Matrix Algebra Simplification

Applying the results of the last section leads immediately to a very simple computational scheme for b**. Starting with

$$\text{I]}\quad b^{**}(p) = (H + pN)^{-1}(Hb + pNb^*)$$

and rewriting H and N as

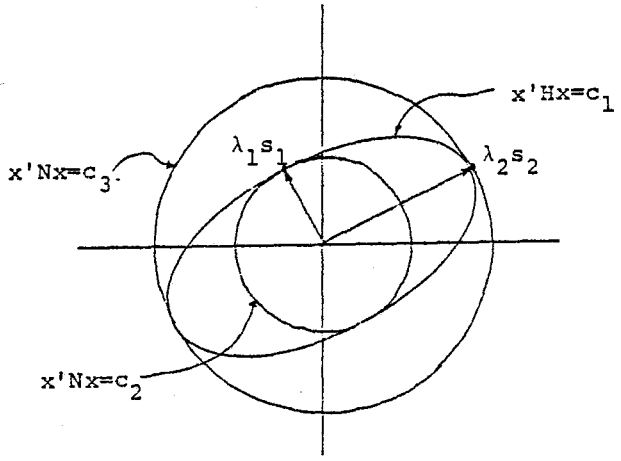
$$H = S'^{-1} I S^{-1}$$

$$N = S'^{-1} D S^{-1}$$

leads, on substitution, to

$$\begin{aligned} b^{**}(p) &= (S'^{-1}IS^{-1} + pS'^{-1}DS^{-1})^{-1} \\ &\quad (S'^{-1}IS^{-1}b + pS'^{-1}DS^{-1}b^*) \\ &= (S'^{-1}[I+pD]S^{-1})^{-1} \\ &\quad (S'^{-1}[IS^{-1}b + pDS^{-1}b^*]) \\ &= S[I+pD]^{-1}S'S'^{-1}(S^{-1}b + pDS^{-1}b^*) \end{aligned}$$

Figure 1



Geometry of the joint diagonalization of H and N

Setting $\underline{b} = S^{-1}b$ and $\underline{b}^* = S^{-1}b^*$ gives

$$\text{II]}\quad b^{**}(p) = S(I+pD)^{-1}(\underline{b} + pD\underline{b}^*)$$

The computational convenience of computing b** for a specified p using II relative to that using I is enormous. To evaluate I, the kxk matrix inverse $(H+pN)^{-1}$ must be computed. To evaluate II, only the inverse <u>of a diagonal matrix</u> $(I+pD)^{-1}$ must be computed. While it is true that S must also be computed, this need only be done once, and thereafter generating additional points b**(p) requires only a small amount of arithmetic, several orders of magnitude less computation than evaluating I, even when k is only 10 to 20.

c. A Polynomial Representation

II has provided an affordable means of

computing particular points b**(p) by using a joint diagonalization of H and N to eliminate the need to compute kxk matrix inverses for each p. For many purposes, II gives a suitable computational scheme. For others, however, it is still too cumbersome. Happily, it can be reduced to a completely different kind of representation, as a polynomial function, and this permits application of the enormous collective knowledge concerning real valued polynomials with real coefficients. The task of this section is to develop and then scrutinize a polynomial reformulation of II.

Expanding II gives

$$b^{**}(p) = S \begin{pmatrix} 1/1+pd_1 & & & \\ & 1/1+pd_2 & & \\ & & - & \\ & & & 1/1+pd_k \end{pmatrix} \begin{pmatrix} \underline{b}_1+pd_1\underline{b}_1^* \\ \underline{b}_2+pd_2\underline{b}_2^* \\ - \\ \underline{b}_k+pd_k\underline{b}_k^* \end{pmatrix}$$

$$= S \begin{pmatrix} \frac{\underline{b}_1+pd_1\underline{b}_1^*}{1+pd_1} \\ \frac{\underline{b}_2+pd_2\underline{b}_2^*}{1+pd_2} \\ \vdots \end{pmatrix}$$

Now $$\frac{\underline{b}_i+pd_i\underline{b}_i^*}{1+pd_i} = \frac{\prod_{j=1}^{k}(1+d_jp)(\underline{b}_i+pd_i\underline{b}_i^*)}{\prod_{j=1}^{k}(1+d_jp)(1+d_ip)}$$

$$= \frac{\prod_{j\neq i}(1+d_jp)(\underline{b}_i+pd_i\underline{b}_i^*)}{\prod_{j}(1+d_jp)}$$

Both the numerator and the denominator in this expression are simply k^{th} degree polynomials in p. Let

$$Q_0(p) = \prod_{j=1}^{k}(1+d_jp) = \sum_{j=0}^{k} q_jp^j$$

$$Q_i(p) = \prod_{j\neq i}^{k}(1+d_jp)(\underline{b}_i+pd_i\underline{b}_i^*) = \sum_{j=0}^{k} q_{ij}p^j$$

The coefficients of these polynomials, q_j j=0,k and q_{ij} i=1,k j=0,k, can easily be computed from D, $\underline{b}$, and $\underline{b}^*$, and are not themselves functions of p; they are fixed for a particular inference problem in which p is free but b* and N are fixed. Collecting these expressions gives

$$\frac{\underline{b}_i+pd_i\underline{b}_i^*}{1+pd_i} = \frac{Q_i(p)}{Q_0(p)}$$

so that $$b^{**}(p) = S \begin{pmatrix} Q_1(p)/Q_0(p) \\ Q_2(p)/Q_0(p) \\ \vdots \end{pmatrix}$$

$$= (1/Q_0(p))S \begin{pmatrix} Q_1(p) \\ Q_2(p) \\ \vdots \end{pmatrix}$$

Now $$\begin{pmatrix} Q_1(p) \\ \vdots \\ Q_k(p) \end{pmatrix} = \begin{bmatrix} q_{10} & q_{11} & q_{12} & \cdots & q_{1k} \\ q_{20} & q_{21} & q_{22} & \cdots & q_{2k} \\ & & \vdots & & \\ q_{k0} & q_{k1} & q_{k2} & \cdots & q_{kk} \end{bmatrix} \begin{bmatrix} 1 \\ p \\ \vdots \\ p^k \end{bmatrix}$$

$$= Q \begin{pmatrix} 1 \\ p \\ p^2 \\ \vdots \\ p^k \end{pmatrix}$$

$$= Q \begin{bmatrix} 1/q_0 & & & \\ & 1/q_1 & & \\ & & \cdot & \\ & & & 1/q_k \end{bmatrix} \begin{bmatrix} q_0 & & & \\ & q_1 & & \\ & & \cdot & \\ & & & q_k \end{bmatrix} \begin{bmatrix} 1 \\ p \\ \cdot \\ p^k \end{bmatrix}$$

Letting $$SQ \begin{bmatrix} 1/q_0 & & & \\ & 1/q_1 & & \\ & & \cdot & \\ & & & 1/q_k \end{bmatrix} = A = \begin{bmatrix} [a_0] & & & \\ & [a_1] & & \\ & & \cdots & \\ & & & [a_k] \end{bmatrix}$$

gives

$$b^{**}(p) = [1/Q_0(p)][q_0a_0+q_1a_1p+\ldots+q_ka_kp^k]$$

and finally, expanding $Q_0(p)$,

$$\text{III]} \quad b^{**}(p) = \sum_{j=0}^{k} q_ja_jp^j \Big/ \sum_{j=0}^{k} q_jp^j$$

where q_j is the coefficient of p^j in $\prod_{m=1}^{k}(1+d_m p)$

q_{ij} is the coefficient of p^j in $\prod_{m\neq i}^{k}(1+d_m p)(\underline{b}_i+pd_i\underline{b}_i^*)$

and $a_j = S\begin{pmatrix} q_{1j}/q_j \\ q_{2j}/q_j \\ \vdots \\ q_{kj}/q_j \end{pmatrix}$

The computational simplicity of III should be apparent. [4] Computing b**(p) for a specified p is a matter of taking a polynomial weighted average of the points a_j j=0,k. The arithmetic involved is of roughly the same order as that required for evaluation of II. The computations in III may be less accurate numerically because of the wide range of powers of p even when k is as small as 10 or 20. Thus, the importance of III relative to II is not in computational simplicity or accuracy. Rather, as mentioned before, it is in the fact that using III in problems involving functions of b**(p) permits the application of the enormously rich known structure of polynomial functions. The advantages of this will become more clear later, when the discussion returns to the issues of characterizing changes in b**(p). In the remainder of this section, additional structure implicit in III will be developed.

The endpoints of the curve traced by b**(p) are easy to identify. From

I] $b^{**}(p) = (H+pN)^{-1}(Hb+pNb^*)$

it is immediate that

$b^{**}(0) = b$

and $\lim_{p\to\infty} b^{**}(p) = b^*$

so long as both H and N are of full rank. Since III is equivalent, it must give the same endpoints, so that

$b^{**}(0) = a_0 = b$

and $\lim_{p\to\infty} b^{**}(p) = a_k = b^*$

Thus, when H and N are of full rank, the first and last "ideal points" are the data and prior means, respectively.

The way in which the weighting of the a_i changes as p increases is also illuminating. As just mentioned, the weights at p=0 and as p becomes large are 1 for the data and prior points respectively, and 0 for the other points. Evidently, as p increases the weight on a_0 gradually falls and the weight on a_k increases. Define

$$w_i = w_i(p) = q_i p^i / \sum_j q_j p^j$$

so that $b^{**}(p) = \sum_i w_i(p) a_i$

and $\sum_i w_i = 1$, $w_i \geq 0$ for i=0,k. The statements about endpoints made above amount to the observation that $w_0(0) = 1$ and $\lim_{p\to\infty} w_k(p) = 1$.

A simple derivation reveals that

$$\frac{\partial w_i(p)}{\partial p} = \frac{w_i}{p}\left[\, i - \sum_j j w_j \,\right]$$

Thus, $w_i(p)$ is increasing if its index i is greater than the current "average index" $\sum_j j w_j$. This average index is 0 for p=0 and approaches k as p becomes large. Letting

$$n(p) = \sum_j j w_j(p)$$

then

$$\frac{\partial n(p)}{\partial p} = \sum_j j \frac{\partial w_j(p)}{\partial p}$$

$$= (1/p)\left[\sum_j j^2 w_j(p) - \sum_j j w_j(p) \sum_m m w_m(p)\right]$$

$$= (1/p)\left[\, E_w(j^2) - (E_w(j))^2 \,\right]$$

$$= (1/p)\, \mathrm{Var}_w(j) \geq 0$$

Thus n(p) is monotonically increasing in p, and $w_i(p)$ is increasing when i>n(p), is maxi-

mized when i=n(p), and is then decreasing for i<n(p).

CHARACTERIZATION

In the preceeding section reasonably inexpensive schemes were developed for computing values of b**(p) for specified p. In this section, the discussion centers on the application of these techniques to the general problem of characterizing an invesigator's ambiguity concerning the posterior b** corresponding to his prior and the given data. The methods considered include ways of tracing movements along the curve b**(p) as well as shifts in the curve resulting from changing the prior location b*. The curve b**(p), called the "contract curve" or "curve decolletage" [4], can be characterized by

1) tracing the curve, i.e., computing a set of points b**(p) for p= $p_1, p_2, \ldots p_m$;

2) computing turning points in the curve, so that extreme values are not missed;

3) reporting the statistical ambiguity, or variance covariance, of estimates b**(p) at particular values of p; and

4) computing the effects on b**(p) (at various values of p) of shifting the curve by changing the prior mean b*.

These methods are elaborated, and solutions based on the techniques developed in the last section are presented in the sections below. Before turning to these issues, the concept and geometry of the contract curve are developed to motivate the discussion that follows.

In the original derivation of the contract curve I, the formula was derived as a maximization of posterior likelihood conditional on the value of p. This emphasized the role of the investigator's ambiguity about p (from lack of knowledge either about λ or about σ^2) in determining a part of his ambiguity about b**(p). This derivation can also be viewed in a slightly different light that emphasizes the geometry of the curve as a whole. Rather than conditioning on p (or λ), condition instead on the value of the data (resp. prior) likelihood, and think of finding the point that, subject to this constraint, maximizes the prior (resp. data) likelihood. These likelihoods are monotone functions of the quadratic forms $(b-\beta)'H(b-\beta)$ and $(b^*-\beta)'N(b^*-\beta)$, so that the formal problem is

$$\underset{\beta}{\text{minimize}} \quad (b-\beta)'H(b-\beta) \text{ subject to}$$
$$(b^*-\beta)'N(b^*-\beta) = c$$

The first order conditions are

$$2H(b-\beta) + 2pN(b^*-\beta) = 0$$

where p is the Lagrange multiplier for the constraint. Thus

$$b^{**} = (H+pN)^{-1}(Hb+pNb^*)$$

This is, of course, exactly the formula I for the contract curve. Thus, the contract curve is the set of points corresponding to

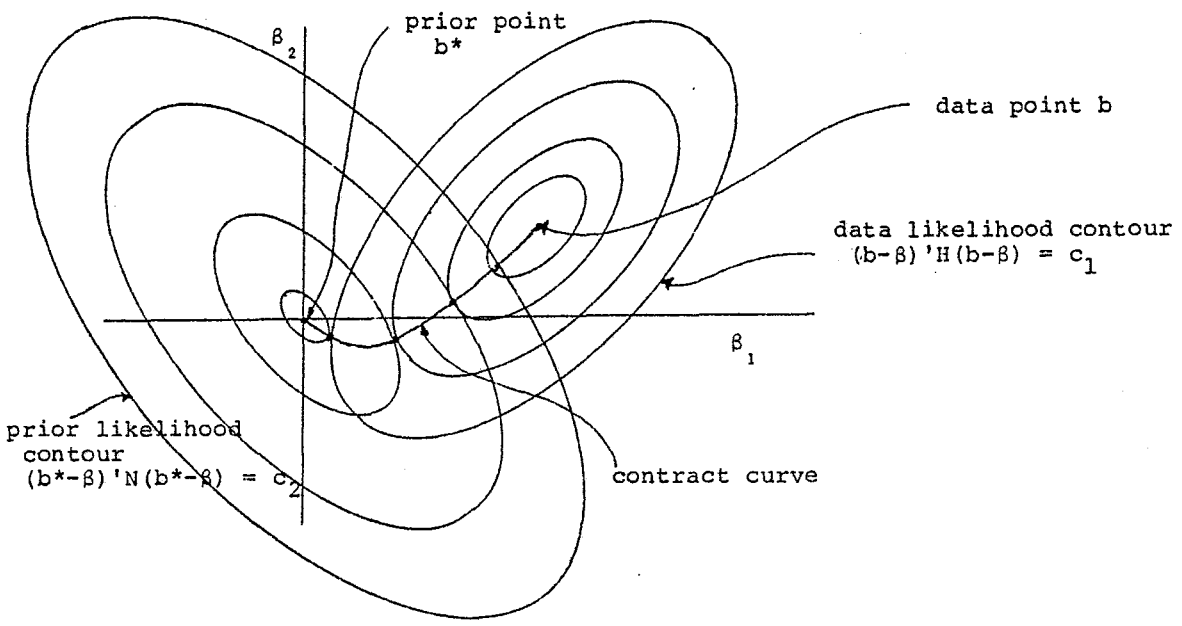
1) the posterior mean conditional on the relative prior and data precision scale factor p; and

2) the highest possible prior (resp. data) likelihood given a set level of data (resp. prior) likelihood.

Figure 2 illustrates the second of these. Subject to a chosen level of data likelihood, the contract curve point is the point at which the prior likelihood contour is tangent to the chosen data likelihood contour. It is this feature that gives the name familiar to economists, the "contract curve."

It should be apparent that the investigator may not be directly concerned with a particular coefficient, but rather, might care about the posterior value of a function of the

Figure 2



Geometry of the contract curve

coefficients. For nonlinear functions there is little that can be said about general characteristics. Linear functions, of course, are as simple to characterize as the coefficients, since the coefficients are special cases of linear functions. In what follows, therefore, the discussion will be couched in terms of characterizing linear functions of the coefficients along the contract curve.

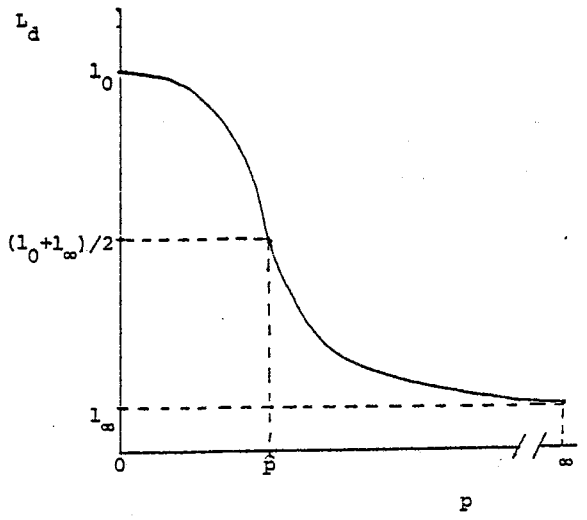
a. Tracing the Contract Curve

The first and most obvious step toward characterizing a function $1'b^{**}(p)$ along the contract curve is to compute the value of the function at a selected set of points $p \varepsilon R = \{p_1, p_2, \ldots p_m\}$. At issue is how to select an interesting set of points R. Two sets have been found generally useful: 1) the set of p's corresponding to $\lambda = 2^{\pm m}\lambda^*$, where λ^* is the investigator's best guess for λ; and 2) the set determined by taking equal steps in terms of data (or prior) likelihood from the data to the prior.

The first set is computed easily by evaluating II or III for the indicated values of p. It is useful becuase it is related to the investigator's best guess about the strength of his prior. It leads to statements of the form "if my prior precision were twice as great, the posterior would move to..." Thus, it gives an insight into the sensitivity of the posterior to the strength of belief in the prior means.

The second set is not so easy to compute, because the values of p that yield the interesting points are specified indirectly. Figure 3 shows a hypothetical plot of the data likelihood function along the contract curve as parameterized by p. The investigator might want to know the point on the contract curve that is halfway (in terms of data likelihood) from the data to the prior. [5] This

Figure 3



Data likelihood function along the contract curve as parameterized by p. $\hat{p}$ is "half way" from data point (p=0) to prior point (p=∞) in terms of data likelihood.

would have a data likelihood $(l_0 + l_\infty)/2$, and would correspond to $\hat{p}$. Whether the stated criterion for selection of points is in terms of the level of the likelihood function, as in this example, or in terms of the F value for the test of the point against the least squares point, the criterion can be translated into a value for the quadratic form $(b-\beta)'H(b-\beta)$ for the data likelihood or $(b^*-\beta)'N(b^*-\beta)$ for the prior likelihood. Taking the example of a criterion stated in terms of the data likelihood, the problem thus becomes

find $\hat{p}$ and $b^{**}(\hat{p})$ such that

$$(b^{**}(\hat{p})-b)'H(b^{**}(\hat{p})-b) = c$$

where c is determined by the selection criterion. Expanding this in terms of I gives

$$c=(((H+pN)^{-1}(Hb+pNb^*) - b)'H(((H+pN)^{-1}(Hb+pNb^*) - b)$$

a single equation in one variable (p), but gives no insight into how one might solve for p. In fact, it looks like a computational nightmare. It is here that the polynomial expansion is of such great advantage. Expanding in terms of III gives

$$c= (\Sigma_i w_i(p)a_i - b)'H(\Sigma_i w_i(p)a_i - b)$$

or, since $\Sigma_i w_i = 1$,

$$c= (\Sigma_i w_i(p)(a_i-b))'H(\Sigma_i w_i(p)(a_i-b))$$

Letting $z_{ij} = (a_i-b)'H(a_j-b)$ and substituting gives

$$c= \Sigma_i \Sigma_j w_i(p)w_j(p)\ z_{ij}$$

Substituting for the w's leads to

$$c \Sigma_l \Sigma_m q_l q_m p^{l+m} = \Sigma_i \Sigma_j q_i q_j p^{i+j} z_{ij}$$

and finally

$$P(p) = \Sigma_i \Sigma_j q_i q_j (z_{ij} - c) p^{i+j} = 0$$

The polynomial on the left hand side, P(p), is simply a real valued polynomial of degree 2k with real coefficients, defined on the positive real line. P(p) is the difference at p between $(b^{**}(p)-b)'H(b^{**}(p)-b)$ and c. Since $b^{**}(0)=b$, $P(0)= -c$ and P is a monotonically increasing function of p, so that P(p) can have at most one positive root. Of course, the solution p^* for which $P(p^*)=0$ could be arbitrarily large, but in practice there is a (reasonably small) value $\underline{p}$ for which any larger value of p gives $b^{**}(p)$ essentially indistinguishable from $\lim_{p\to\infty} b^{**}(p) = b^*$. Thus, for practical purposes the domain of P is limited to $(0,\underline{p})$. Since P is monotone, a simple interval splitting algorithm can be used to solve for p^*. For a particular criterion, then, the procedure is as follows:

1) identify the values c_i of the criterion that yield interesting points,
2) for each c_i, solve $P_{c_i}(p)=0$ for p_i, and
3) using either II or III, evaluate $b^{**}(p_i)$

It should be apparent that the polynomial expansion is quite useful in solving a variety of problems of this general type. Again, the relevant feature is the reduction of a complex expression in matrix algebra to a simple polynomial roots problem.

b. Turning Points on the Contract Curve

The procedures just discussed for characterizing the contract curve amount to choosing, in advance, a set of points at which to observe the values b**(p) or l'b**(p). In most examples, the curve is sufficiently smooth that this does not obscure extreme values of the chosen linear functions. To guard against the possibility that some bend in the curve might be entirely missed by the "grid" procedures discussed above, it seems appropriate to compute the turning or extreme values of l'b**(p) along the curve. Defining

$$f(p) = l'b^{**}(p)$$

the problem is to find the set of points p* for which f'(p*)=0. Using the polynomial expansion III,

$$f(p) = l'\left(\sum_i w_i a_i\right)$$

$$= \sum_i w_i \, (l'a_i)$$

Letting $l_i^* = l'a_i$ gives

$$f(p) = \sum_i l_i^* w_i(p)$$

so that

$$f'(p) = \sum_i l_i^* \frac{\partial w_i(p)}{\partial p}$$

$$= \sum_i l_i^* \frac{w_i}{p} \left(i - \sum_j j w_j \right)$$

Thus

$$pf'(p) = \sum_i l_i^* w_i i - \sum_i l_i^* w_i \sum_j j w_j$$

Since $\sum_i w_i = 1$,

$$pf'(p) = \sum_j w_j \sum_i l_i^* w_i i - \sum_i l_i^* w_i \sum_j j w_j$$

$$= \sum_i \sum_j l_i^* (i-j)(w_i w_j)$$

Ignoring the values p=0 and the limit as $p \to \infty$, which are defined as extreme points because they are endpoints, the extreme values of f(p) occur at points at which f'(p) or pf'(p) are 0. Thus, the extreme points correspond to values for which (expanding the weights w_i)

$$\sum_i \sum_j l_i^* (i-j) q_i q_j p^{i+j} \Big/ \left(\sum_m q_m p^m \right)^2 = 0$$

or, more simply,

$$Q(p) = \sum_i \sum_j l_i^* (i-j) q_i q_j p^{i+j} = 0$$

Q(p) is a real valued polynomial of degree 2k with real coefficients. It need have no, and may have up to 2k positive real roots. There are no obvious properties of Q that can be exploited to make its roots easy to find, but in practice little difficulty has been encountered in using readily available general polynomial roots algorithms. Once more, it should be clear that it is the transformation of the matrix weighted average problem to a polynomial roots problem that permits its easy solution.

c. Statistical Ambiguity of the Posterior

In addition to the ambiguity of the posterior estimate that comes from the lack of certainty about the prior, there is the purely statistical ambiguity engendered by the random errors in the data and the prior. [6] The usual characterization is in terms of the standard error of the estimate. Conditional on a value of p, and presuming that s^2 is a correct estimate of σ^2, the variance covariance matrix of b**(p) is

$$V(b^{**}(p)) = (H+pN)^{-1} = S(I+pD)^{-1}S'$$

as shown in the derivation of II. Thus, the

variance of $f(p) = 1'b^{**}(p)$ is given by

$$V(f(p)) = 1'S(I+pD)^{-1}S'1$$

Letting $\underline{1} = S'1$, the standard error (conditional on p) is simply

$$SE(f(p)|p) = \sqrt{\sum_i \underline{1}_i^2 / (1+pd_i)}$$

Once S and S'1 are computed and p is specified, this is a trivial computation. It should be noted that what has been exploited here is the reduction of H+pN to a product of fixed matrices and a readily invertible diagonal matrix I+pD.

d) Sensitivity to Changes in the Prior Mean

In the preceeding discussion, various characterizations of a fixed contract curve have been given. In essence, these amount to statements about movements along a given contract curve. It remains to characterize how the curve itself would shift if the prior on which it is based were changed. The discussions above have covered one kind of shift, changes in the scale of prior precision, λ. It is clear that changes in the structure of prior precision, N, would be very expensive to deal with, since all of the simplifications described have come from the joint diagonalization, which would be altered if N were changed. Thus, the remaining shifts that can be readily handled are those that come from changing the prior means b*. Starting with

$$b^{**}(p) = S(I+pD)^{-1}(S^{-1}b+pDS^{-1}b^*)$$

which was encountered in the derivation of II gives

$$\frac{\partial b^{**}(p)}{\partial b^*} = S(I+pD)^{-1}pDS^{-1}$$

and similarly

$$\frac{\partial 1'b^{**}(p)}{\partial b^*} = 1'S(I+pD)^{-1}pDS^{-1}$$

These expressions are simple functions of p, easily computed once S and D are known. Note that the derivative at p=0 is 0; when the prior is given no weight, changes in the prior mean do not matter. As $p\to\infty$, the derivative approaches 1'; as all the weight is given to the prior, the posterior moves exactly as the prior mean moves. It should also be noted that these expressions do not depend on b*; the posterior (conditional on p) is linear in the prior mean, so that these derivatives give exact measures of the changes in the posterior engendered by alterations in the prior mean. The role of the joint diagonalization in facilitating all of these computations should be apparent.

CONCLUSION

This paper has dealt with a series of characterizations of an example of a matrix weighted average that arises commonly in Bayesian econometrics. A set of computational simplifications of the basic formula of the matrix weighted average was developed. The use of these tools was then illustrated by applying them to the example of exploring the ambiguity in the posterior for a set of coefficients of a linear model with normal errors. The ambiguity arose first because of uncertain knowledge of the normal prior, and second from the purely statistical errors in the data and prior. A number of insights into the relation between the posterior and the prior were explored.

Much work remains to be done. First, it would be helpful to have better algorithms for jointly diagonalizing positive definite matrices. Second, better polynomial root

routines would facilitate some of the computations made feasible by the polynomial representation. Finally, the relation between the joint diagonalization of $(N+\delta N, H)$ and that for (N,H) needs to be explored in more detail. The joint diagonalization is at the heart of all the simplifications discussed in this paper; if a way could be found to use the information from one diagonalization in computing another, then the final source of ambiguity, uncertainty about the prior precision structure N, could be explored more systematically.

NOTES

[1] One could specify a more general (Student's τ) prior that included uncertainty about σ^2. This complicates the computations immensely without altering the interpretability of the results. The expedient of examining posteriors conditional on p is chosen because it is simpler in computation and is at least as simple intuitively.

[2] Two approaches come to mind as avenues for investigating the effects of changing N. First, one could compute the kinds of characterizations of the effects of ambiguity in λ and b^* for various different matrices N. Second, one could adopt the extreme view that any N is possible and compute bounds for the possible locations of b^{**}. c.f. Chamberlain and Leamer [1 page 78].

[3] In all of what follows it suffices that H and N both be positive semi definite (psd) and that H+N be positive definite (pd). For the sake of simplicity, H will always be taken to be of full rank.

[4] The derivation just given amounts to a constructive proof of a theorem presented by Leamer and Chamberlain [2 page 88]. They also give other ways to compute the a_i, which they refer to as "rotationally invariant" or "ideal" points.

[5] In the SEARCH package [3] the contract curve is presented in terms of 10 equal steps of data likelihood.

[6] Most investigators appear to think about this problem the other way, i.e., that the statistical ambiguity is large relative to the ambiguity from the prior. There appears to be no good justification for this widely held belief.

REFERENCES

[1] Chamberlain, Gary and Edward E. Leamer, "Matrix Weighted Averages and Posterior Bounds," Journal of the Royal Statistical Society, Series B Volume 38 Number 1, 1976, pp 73-84.

[2] Leamer, Edward E. and Gary Chamberlain, "A Bayesian Interpretation of Pretesting," Journal of the Royal Statistical Society, Series B Volume 38 Number 1, 1976, pp 85-94.

[3] Leonard, Herman B., "SEARCH: Manual for Bayesian Inference," Harvard Institute for Economic Research Technical Paper Number 14, September, 1977.

[4] Lindley, D.V. and A.F.M. Smith, "Bayes Estimates for the Linear Model," Journal of the Royal Statistical Society, Series B Volume 34 Number 1, 1972, pp 1-41.

A standard text on the algebra of matrices, such as Gantmacher, Theory of Matrices, may be useful in reading this paper.

MODULARITY, READABILITY, AND TRANSPORTABILITY OF THE COMPUTER-ASSISTED DATA ANALYSIS (CADA) MONITOR

Melvin R. Novick, Gerald L. Isaacs, and Dennis F. DeKeyrel
The University of Iowa

ABSTRACT

The Computer-Assisted Data Analysis (CADA) Monitor is a conversational interactive computer system designed to enable educational researchers and administrators, some of whom may be relatively inexpert in statistical methods to analyze data more expertly through the use of Bayesian statistical and exploratory data analysis methods. In designing the Monitor three considerations were taken into account in order to maximize the Monitor's availability to users with varying degrees of statistical expertise and hardware capabilities. The three design considerations are modularity, readability, and transportability. First, to reduce programming costs and to facilitate the incorporation of advances in numerical analysis techniques, the CADA Monitor is constructed from building blocks that are chained or appended whenever a particular computation is needed in any analysis. Second, to insure that the user has available the information and instructions necessary to carry out the step-by-step analysis, every effort is made to present the material as clearly and completely as possible. Third, to make the CADA Monitor available on as many different hardware configurations as possible, the programs are written in a subset of the BASIC programming language that is common to major dialects of the language.

I. CADA: An Introduction

The Computer-Assisted Data Analysis (CADA) Monitor is a conversational interactive computer system designed to enable educational researchers and administrators, some of whom may be relatively inexpert in statistical methods, to analyze data more expertly through the use of Bayesian statistical and exploratory data analysis methods. The CADA Monitor is intended for two audiences. On the one hand, it has as its primary function the teaching of Bayesian statistical and exploratory data analysis methods to students with minimal mathematical background. On the other hand, it provides educational researchers, administrators, and others with easily used, yet sophisticated methods of exploratory and confirmatory statistical analyses.

CADA leads a user through a statistical analysis on a step-by-step basis so that he must "do things correctly." It uses and teaches only correct procedures. The CADA system makes use of the two major strengths of the computer -- computational speed and memory -- as adjuncts to the human thought process and makes these two capabilities more directly usable than do previous systems. At each decision point, the computer accomplishes the necessary calculations, which can be of varying complexity, and presents to the user all data necessary for the particular decision to be made at that time.

In constructing the Monitor, we made several assumptions about the optimal man-machine interaction. It was first assumed that data analysis will be made easier if the user never has to rely on his own memory. That is to say, that at any given point in the analysis the frame which the user sees must contain all information that he will need for the decision that he now must make. Second, it was decided that the same system should be made available to all users regardless of their level of ability. This approach differs from that taken in other systems which have the capability of providing various levels of prompt for users with dirrerent amounts of familiarity with the system or expertise in the statistical method. We have decided to use a single mode of presentation, having judged that it is difficult for even the user himself to know precisely what level of prompt he may need at a given time. Thus a user may be quite sophisticated but may have a lapse at a particular point and want additional information.

The third assumption made in the design of CADA is that it is important in the conversational man-machine interface for the expert to be able to enter or re-enter an analysis at any point he wishes. He must be able to back up and redo a portion of the analysis without having to redo those parts of the analysis he wishes to retain. In designing the system as we have, it has been our intent to give the user the feeling that the computer is working for him rather than that he is working for the computer.

The fourth assumption is that user input should always be numerical. Yes-no switching is done by 1-0 coding. Judgments are entered numerically as scalars or vectors. The primary advantage of this approach is the ease of user input. A second advantage is that CADA can be run with very inexpensive user terminals (current cost about $1,000) and very inexpensive computers (current cost as low as $12,500), and indeed there should be no difficulty in using CADA with a touch-tone telephone as an input device, a modified TV set as display, and a $6,000 LSI-11 microprocessor-based personal computer with an ANSI level one semi-compiled BASIC and an extended arithmetic floating point chip.

A fifth assumption is that CADA will grow to encompass additional statistical procedures as time and resources permit. Yet we know that many statistical computations are common to a variety of analyses. Thus it seems desirable to relate one analysis to another and borrow subroutines in constructing new programs.

These basic assumptions led us to three major considerations in designing the Monitor. The first of these is modularity. The CADA Monitor is constructed from building blocks that are chained or appended whenever a particular computation is needed in any analysis. The second is readability, i.e., that the material presented be as clear, complete, and accessible as possible. A third major design consideration is transportability, since we hoped to make the system available on as wide a variety of machines as possible. Each of these considerations will now be discussed in detail.

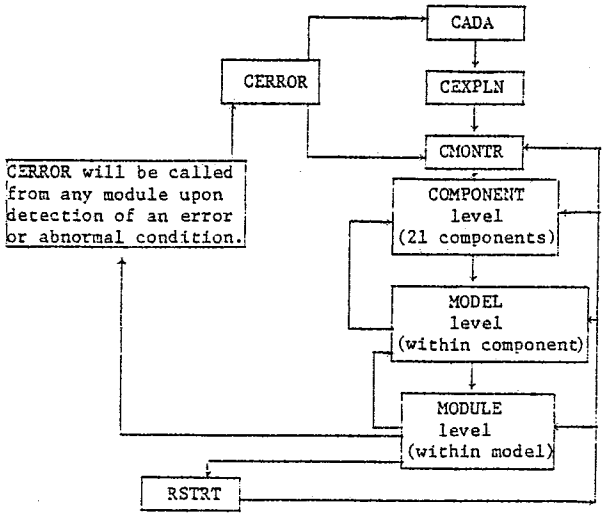
II. Modularity

The central design strategy of the CADA Monitor is modularity. Whenever a statistical routine is constructed the required steps are broken down into logical steps that are programmed separately as building blocks to be used in the present program and possibly in future programs. Thus now, after five years of development, an organized catalogue of subroutines that can be appended during program construction has been collected. The savings in programming time is obvious. However, a second benefit may not be so obvious.

Numerical analysis techniques have improved substantially during the past decade and further significant improvements are continuing to be made. Because CADA is so highly modularized any improvement in one building block subroutine automatically benefits all programs in which it is used with little additional labor being required to modify these programs.

The following, in outline form, is a description of the supervisory routines, components, models, and modules which comprise the CADA Monitor. Figure 1 provides an overview schematic diagram of the structure of the Monitor.

Figure 1



There are five supervisory routines that monitor the presentation of programs to the user. The following is a listing and brief explanation of the function of each subroutine.

A. CADA -- Initializes the Monitor, chooses and zeros restart file, zeros appropriate constants, sets form feed appropriate to the terminal being used, passes control to CEXPLN.

B. CEXPLN -- Gives a brief explanation of the workings of the Monitor, passes control to CMONTR.

C. CMONTR -- Sets restart values in first record of restart file. Leads the user through component, model, and module levels of the analysis the investigator wants by passing control to appropriate module.

D. CERROR -- Gives instructions in case of unexpected failure or user-initiated break. Passes control either to CADA (on user-initiated break) or CMONTR (otherwise).

E. RSTRT -- Sets values in restart file to correspond to user's new choice of next step in his analysis. Passes control back to CMONTR.

There are three distinct levels in the structure of the Monitor: component; model; module. The highest level is called the component level. The fifteen different available components are summarized as:

COMPONENT 1: Data Management
COMPONENT 2: Data File Catalogue
COMPONENT 3: Probability Distributions
COMPONENT 4: Binary Models
COMPONENT 5: Univariate Normal Models
COMPONENT 6: Multi-Category Models
COMPONENT 7: Simple Regression and Correlation
COMPONENT 8: Utilities and Expected Utilities
COMPONENT 9: Simultaneous Estimation of Proportions
COMPONENT 10: Simultaneous Estimation of Means
COMPONENT 11: Multiple Regression and Correlation
COMPONENT 12: Simultaneous Prediction in m-Groups
COMPONENT 13: Educational and Employment Selection
COMPONENT 15: One-Way Model I ANOVA
COMPONENT 16: Multiway Model I Factorial Designs

All components share the same general structure, with the exception of Components 1, 2, and 3. Components 1, 2, and 3 are slightly different in that they only have one level under them. Component 1 contains data management procedures, Component 2 contains the data file catalogue, and Component 3 contains routines for evaluating standard probability distributions. In some ways Component 3 may be thought of as the core of CADA. It contains programs for doing most of the non-trivial computations required in statistical analyses. The following is a list of the distributions currently available on CADA:

1. Normal
2. Student's t
3. Inverse Chi
4. Inverse Chi-Square
5. Chi-Square
6. Beta
7. Behrens-Fisher
9. Snedecor's F
10. Binomial
11. Pascal
12. Beta Binomial
13. Beta Pascal
14. Poisson
15. Gamma

All other components are constructed as follows: Each component has a set of models associated with it. There are usually several models for each component. For example, Component 4, Binary Models, consists of:

1. Beta Binomial Model
2. Mixed Beta Binomial Model (not yet available)
3. Beta Pascal Model
4. Mixed Beta Pascal Model (not yet available)
5. Comparison of Two Proportions
6. Gamma Poisson Model (not yet available)

Each model has a set of modules under it. These modules are the actual data analysis programs. For example, the Beta Binomial Model has five modules:

1. Prior distribution on proportion (PI)
2. Preposterior analysis
3. Posterior distribution on PI
4. Posterior intervals for PI
5. Predictive intervals

At the module level, sequencing is automatic under the control of the Monitor. The analyst is led by the Monitor, module by module, through the model he has selected.

There exists a restart capability which allows the user to jump from module to module without loss of data. The user may also select a different model within the component in which he is now operating or to any other component. Any time the user is asked for input he may type -9999 and enter the restart module RSTRT. He is then given the following choices:

1. Restart at the beginning of this module.
2. Select a different module from this model.
3. Select a different model from this component.
4. Select a different component.
5. Exit CADA Monitor.

This arrangement permits the user to redo only a small part of the analysis and to retain the larger part with which he may be satisfied.

Progress of a user through an analysis is directed by the CADA Monitor, which takes him from step to step, but always lets him retrace his steps. The Monitor guides, but does not control the analysis.

Since many mathematical manipulations are the same from module to module, CADA's data-analysis routines have been built in a subroutine manner. A large percentage of the computation done by CADA involves the use of computational subroutines. The subroutines have been assigned fixed line numbers and are appended to the appropriate modules. The following is a list of these subroutines. The line numbers listed here are those assigned to the subroutines on the HP 2000 ACCESS version; other versions may or may not conform to these assigned line numbers. Note that these line numbers are fixed to allow easy utilization.

Subroutine	Description	Line Number
BDTR	Beta Distribution CDF	5000
BHDR	Beta HDR	7000
ICHDR	Inverse Chi HDR	7300
CSQHDR	Chi-Square HDR	6600
ICSQHR	Inverse Chi-Square HDR	7800
TDTR	Student's t Distribution CDF	6000
THDR	Student's t HDR	7500
NDTR	Normal CDF	8000
LGAM	Log Gamma	5850
CSQDTR	Chi-Square CDF	5500
INPRT1	One Parameter Input	9000
INPRT2	Two Parameter Input	9050
INPRT3	Three Parameter Input	9100

III. Readability

The approach to readability taken in CADA is, we believe, the correct one in its context. It is not necessarily correct for other applications. First we note that the CADA project is committed to lowering the cost of computation, making computation available to locations with minimal capital resources, and providing this capability to institutions having a wide variety of computer hardware, including those having only minicomputers and microprocessors.

These considerations rule out any possibility of providing an artificial intelligence-based dialogue with user-generated verbal input being syntactically or semantically parsed to provide instruction to the computer. Clearly such an approach to statistical computation is desirable and attainable, however we would view our current work as being compatible with that approach. Any artificially intelligent reading of a user's input involving a semantic grammar will require a modularized response capability under the control of a monitor or mediator similar to that provided by CADA.

A common second approach that we have not adopted involves making available two or three levels of textual accompaniment of questioning, depending upon a prespecified level of expertise of the user. This approach usually also incorporates a help call which provides additional text. Our own use of these techniques convinces us of our inability to decide just how much text is necessary for each question for each of the levels of expertise. We also found that there is much variability in need for help among users. Our solution to the problem, therefore, is to present a full screen of information when asking a question. However, this screen is highly formatted so that the expert can pick out the information he needs without reading everything on the screen. This, of course, presupposes fast screen-writing and a low cost for text transfer, both of which can be achieved with minicomputers and microprocessors. With larger machines the text transfer can be expensive, depending on the charge schedule of the individual computer.

With respect to charges, a brief parenthetical remark might be added. For CADA, the concept of economy of scale in selecting a computer is fallacious. For CADA, the larger the computer, the greater the cost. Use of a CDC CYBER might average out to an all-inclusive charge of $10 or more an hour for CADA, whereas use of an LSI-11 based microprocessor might be $1. Either figure, however, represents an incredible bargain.

Regardless of what approach is taken to readability, we believe that the central concept should be the assumption of zero memory for the user. As Sherlock Holmes pointed out, brain cells used to retain facts cannot be used to process information. Indeed, the great virtue of the computer is that it can store facts and retrieve them instantaneously, thus freeing the mind to think. CADA is organized to emphasize fully the use of this strength of the computer.

IV. Transportability

The most important development in the computer industry during recent years has been the rapid decrease in the cost of computers. This is true for central processors, memory, mass storage, peripherals, and terminals. Recent figures show that the price/performance ratio of hardware has improved by a factor of 100 each decade since 1955 and indications are that this will continue at least for the next decade (Computer World, August 8, 1977, p. 1).

As a result of this trend, a purchaser of today's computer hardware can buy more than three times as much power for one-third the price of a computer five years ago.

While the cost of hardware has dropped dramatically, the cost of computer software has risen substantially over the past few years and will continue to do so in the future. This is especially true in the area of minicomputers and microprocessors, where new software is needed and must currently be amortized over a small base. However with the increasing power and capabilities of minicomputers and microprocessors, there is a growing demand for new packages and this demand has not been met by the manufacturers or by software houses. In order that the gains in the hardware capability, particularly in microprocessors, be effectively translated into all-around increased computer effectiveness, the need for software packages must be met and at a decreased cost.

One means of counteracting the shortage and high cost of software is for users to pool and share their packages. Most hardware manufacturers have formed users groups and in the educational context there is an NSF-funded group, CONDUIT, based at the University of Iowa, whose purpose is to distribute computer software. However, in order for software to be shared by many users it must be transportable. This means that it must be written in a language that is supported on a wide variety of machines and also that the language is compatible from machine to machine. Currently the only language that fills these requirements is BASIC.

Unfortunately, the situation is complicated by the fact that BASIC has many dialects that may differ in both syntax and semantics. Different machines from the same manufacturer may not even have the same dialects of BASIC. For example, the Digital Equipment Corporation has one version of BASIC for its PDP-10 and two entirely different versions for its PDP-11. However, it has been shown that if a few simple rules are followed when a program is written, it can be transported from machine to machine with a minimum of time and trouble (Isaacs, 1973, 1976).

With the first designs of CADA, it was anticipated that it would be widely used on a variety of different machines. Therefore, CADA was written in such a way as to make transportability from machine to machine relatively straightforward and easy. The CADA Monitor has been written following these rules and transported to a wide

variety of different machines ranging from a Wang 2200 to the large CDC CYBER systems. CADA is now available in the following dialects of BASIC:

1. HP 2000 ACCESS
2. DEC PDP-11 RSTS
3. DEC PDP-10
4. IBM 370 VS BASIC
5. CDC CYBER NOS
6. UNIVAC 1100 SERIES
7. DEC PDP-11 V03 BASIC (summer 1978)

CADA was developed on an HP 2000 ACCESS system and then translated in stages to the other machines. Since CADA includes over 150 programs averaging approximately 600 lines of code, and 30 data files, the translation to each new system takes two experienced persons approximately two weeks. The following is an outline of the steps that were involved in preparing for and carrying out the translation.

The first step in transporting programs from one system to another took place during the program design phase. At this time it was decided what capabilities of the computer were necessary for CADA. It was decided that the following were necessary for a conversational, modularized package such as CADA:

1. Formatted output
2. Program chaining
3. Data files
4. Limited string handling (substring, length, conversion)
5. Minimum of six digits of accuracy

The rules for transportability were then checked so that only the most transportable forms of statements were used. For example, only integer expressions were allowed to appear as a subscript of an array. This avoided the problem of some systems rounding the subscript while other systems truncate the subscript.

The next step in each translation of CADA was to compile a list of statements used in the HP version of CADA that needed translation to the new version. These were kept to a minimum since only a subset of the HP BASIC statements were used. The following statements usually are modified in each translation:

1. PRINT#
2. READ#
3. PRINT USING
4. IMAGE
5. GO TO . . . OF
6. CONVERT
7. string dimensions
8. substrings and other string functions
9. COM
10. CHAIN

A translator program has been written for each dialect of BASIC for which a version of CADA is to be produced. These programs reside on the HP 2000 ACCESS system. They modify the necessary lines of each module and produce an ASCII file that contains the translated programs.

This ASCII file is then transmitted with the appropriate Job Control Language (JCL) by means of the HP's Remote Job Entry (RJE) to an IBM computer where it is written onto magnetic tape. Each program and file of CADA is a separate file on the magnetic tape. The magnetic tapes for all machines contain no internal labels and have the following formats:

	Record Format	Block Size	Record Length	Trks	Density	Character Codes
DEC PDP-11 RSTS	Fixed Blocked	1600	80	9	800 BPI	ASCII
DEC PDP-10	Fixed Blocked	1600	80	9	800 BPI	EBCDIC
IBM 370	Fixed Blocked	4000	80	9	1600BPI	EBCDIC
CDC CYBER	Fixed Blocked	4000	80	9	1600BPI	ASCII
UNIVAC	Fixed Blocked	1600	80	9	1600BPI	ASCII

The microcomputers such as Wang and DEC-11 V03 do not have magnetic tape drives and must be handled differently. The Wang 2200 was configured with a multi-speed RS232 compatible board which was connected to a terminal with a cassette and the programs and files were transmitted via this cassette. On the DEC-11 V03 the transfer will be via floppy disk which can be written from the IBM 360. After the programs have been received by the new machine, there still must be some hands-on translation done. The translator programs handle about 95% of the statements, but some statements must still be translated manually. The remaining 5% of statements that are not automatically translated are those for which the translator needs more information than is available at the time it encounters the statement. For example, the DEC-10 does not allow an array to be written to a file with a MAT PRINT. In this case a FOR-NEXT loop is needed and the size of the array is not known. A second class of statements not translated are those for which a part of a string is modified using a substring on the left side of an assignment statement. This is not permissible on some systems but can be accomplished using a concatenation operator. The final types of statements requiring special treatment are the PRINT USING and IMAGE statements in which cursor control is specified to keep the cursor at the end of the print line instead of at the beginning of the next line. To accomplish this in the DEC PDP-10, for example, the PRINT USING must be written onto a file and then read back into a string and this string is outputted via a PRINT statement with a semicolon at the end of it. Data files to be transferred are written into ASCII files, one entry per record, and passed to the magnetic tape, cassette, or floppy disk. These are then read by a small program written on the new machine and written out by the same program in the appropriate format for the new machine.

CADA is then thoroughly tested on the new machine to ensure that no errors have been introduced during the translation. This testing phase may take several days of the two-week translation period. A program and file tape or diskette is then created using the method that is the easiest to transfer CADA to other machines of the same type and operating system. There are usually programs provided by the manufacturer which allow easy transfer from one machine to another. These allow all of CADA to be loaded from tape with a few simple commands. If a program is not available, the easiest and most straightforward means is documented and sent to anyone receiving CADA. The new tapes containing the transported programs and files are then duplicated at the University of Iowa and sent to anyone using CADA.

It should be noted that there is an American National Standards Institute (ANSI) committee X3J2 which has been commissioned to standardize the BASIC programming language. It has been meeting for almost four years and has produced ANSI standard minimal BASIC. This is a small nucleus from which extensions will be produced that will standardize further enhancements. Minimal BASIC contains only the following statements: DATA, DEF, DIM, END, FOR, GOTO, GOSUB, IF, INPUT, LET, NEXT, ON, OPTION, PRINT, RANDOMIZE, READ, REM, RESTORE, RETURN, and STOP. The following Implementation Supplied functions were also standardized: ABS, ATN, COS, EXP, INT, LOG, RND, SGN, SIN, SQR, TAB, and TAN. However, more importantly it sets down syntax and semantics that are to be followed in future BASICs. The committee is now working on level-one enhancements to ANSI standard minimal BASIC and plans to have these completed late in 1979. CADA has been developed in a manner consistent with the restrictions to the use of the proposed level-one enhancements. When and if level-one BASIC is standardized and adopted by all manufacturers translation will no longer be necessary for CADA. But even an optimistic estimate would put that at least three years into the future.

In the interim, as more and more of BASIC becomes standardized, it will become easier to transport programs from one environment to another. However, all manufacturers will still offer their own enhancements beyond ANSI level-one BASIC. Programmers will still have to avoid these nontransportable features if they desire transportability. With a little discipline and common sense, most programs may be written so that they may be transported from one machine to another without a total rewrite.

V. Implications

The emphasis on modularity, readability, and transportability that we have discussed here is appropriate for the CADA project. It is unlikely that this same emphasis will be appropriate in quite the same way for other projects. Modularity, however, may be important whenever complex computations are common to two or more programs. At this level, there is nothing new in that the concept of shared subroutines was introduced in the earliest computer languages. However, the use of a subroutine monitor in the conversational mode, as developed for CADA, is new, and perhaps its essential tree-like structure (component, model, module) may be useful in other large scientific systems.

The issue of readability will always be resolved on a case-by-case basis. The choice between highly formatted output, prespecified levels of prompt, help routines, and intelligent conversation will also depend upon the application, the targetted hardware, cost considerations, and the desire or lack of desire for transportability.

Transportability should be a dying issue as the ANSI committee completes its work. But the current illness may be a lingering one and the death of the current multiplicity of dialects of BASIC may not be imminent.

VI. References

French, Nancy "Programmer Productivity Rising Too Slowly: Tanaka," Computer World, August 8, 1977, p. 1;6.

Isaacs, Gerald L. Interdialect Translatability of the BASIC programming language. SIGCUE Bulletin, 1974, 8, 11-22. Also appeared as ACT Technical Bulletin No. 11 (reprinted by CONDUIT, 1975). An earlier version was distributed by CONDUIT, 1972.

Isaacs, Gerald L. BASIC Revisited: An Update to "Interdialect Translatability of the BASIC Programming Language." Iowa City, Iowa: CONDUIT, 1976.

Novick, M. R., Isaacs, Gerald L., and DeKeyrel, Dennis F. Computer-Assisted Data Analysis -- 1977, Manual for the Computer-Assisted Data Analysis (CADA) Monitor. Iowa City, Iowa: Iowa Testing Programs, 1977.

A SURVEY OF NUMERICAL INTEGRATION

Arthur H. Stroud
Department of Mathematics
Texas A&M University
College Station, Texas 77843

Abstract. The most important numerical integration methods for functions of one variable are Gauss quadrature formulas. For the regions most often encountered in more than one variable, namely the n-cube, n-sphere, n-simplex and entire n-space, one can use combinations of one variable Gauss formulas. Integrals in which the integrand is a product of a probability density function times another function are considered. Computer subroutines for the univariate normal and bivariate normal probability density functions are given.

1. Introduction

The purpose of this paper is to survey some important methods for approximating definite integrals. First we will consider functions of one variable. Here the methods have the general form

$$\int_a^b w(x)f(x)dx \approx \sum_{k=1}^{N} A_k f(\nu_k) \qquad (1)$$

where $[a,b]$ is a given interval and $w(x)$ is a given weight function which is nonnegative on $[a,b]$. Next we consider approximations analogous to (1) for functions of more than one variable. In this case the approximations have the form

$$\int\cdots\int_{R_n} w(x_1,\ldots,x_n)f(x_1,\ldots,x_n)dx_1\ldots dx_n \approx \sum_{k=1}^{N} A_k f(\nu_{k,1},\ldots,\nu_{k,n}) \qquad (2)$$

where R_n is a given region in n-space and $w(x_1,\ldots,x_n)$ is a weight function which is nonnegative on R_n. Approximations (1) and (2) are called integration formulas, or alternatively, quadrature formulas for one variable and cubature formulas for more than one variable. The A_k are called the coefficients or weights for the formula and the ν_k or $(\nu_{k,1},\ldots,\nu_{k,n})$ the points or nodes.

If an approximation (1) or (2) is exact (i.e., has zero error) for all polynomials of degree $\leq d$ in the n variables and if there is at least one polynomial of degree $d+1$ for which the approximation is not exact, then the integration formula is said to have degree d.

Reference works on formulas (1) are Krylov [2], Davis and Rabinowitz [1], and Stroud and Secrest [4]. A reference for formulas (2) is Stroud [3].

The most important class of formulas (1) are Gauss formulas. In Section 2 we mention some Gauss formulas which can be used when $w(x)$ is one of several well known univariate probability density functions (pdf's). An important class of formulas (2) are obtained by combinations of various formulas for one variable. These are called product formulas [3; Chap. 2]. In Section 3 we mention several

standard types of product formulas. In particular, we consider product formulas which can be used for two pdf weight functions.

2. Integration Formulas for One Variable

First let us assume that $w(x) \equiv 1$ and that $[a,b]$ is an interval of finite length. Then

$$\int_a^b f(x)dx$$

can be approximated in the following way. Subdivide $[a,b]$ into a number of subintervals, say

$$[a,\ a+h],\ [a+h,\ a+2h],\ldots,[a+(M-1)h,\ b]$$

where $h = (b-a)/M$. Then

$$\int_a^b f(x)dx = \int_a^{a+h} f(x)dx + \int_{a+h}^{a+2h} f(x)dx + \ldots + \int_{a+(M-1)h}^{b} f(x)dx$$

and we can use an elementary approximation on each subinterval. If on each subinterval we use the trapezoidal formula

$$\int_\alpha^\beta f(x)dx \approx \frac{\beta-\alpha}{2}\,[f(\alpha) + f(\beta)]$$

the approximation which results is the repeated trapezoidal formula

$$\int_a^b f(x)dx \approx \frac{h}{2}f(a) + h\sum_{k=2}^{N-1} f(a+kh) + \frac{h}{2}f(b)$$

$$h = \frac{b-a}{N-1} \qquad (3)$$

If we start with Simpson's formula

$$\int_\alpha^\beta f(x)dx \approx \frac{\beta-\alpha}{6}\,[f(\alpha) + 4f(\tfrac{\alpha+\beta}{2}) + f(\beta)]$$

we obtain the repeated Simpson's formula

$$\int_a^b f(x)dx \approx \frac{h}{3}[f(a) + 4f(a+h) + 2f(a+2h) + 4f(a+3h) + 2f(a+4h) + 4f(a+5h) + \ldots + f(b)] \qquad (4)$$

$$h = \frac{b-a}{N-1} \qquad N = \text{any odd integer} \geq 3.$$

Each of the approximations (3) and (4) have the property that if it is computed for a sequence of increasing values of N the approximations converge to the true value of the integral whenever the integral exists. Also they are easy to program for a computer. The disadvantages of (3) and (4) are that they usually have low accuracy. The repeated trapezoidal formula has degree 1 and the repeated Simpson's formula degree 3.

Another approach can be used to find quadrature formulas. Assume we allow all of the ν_k, A_k, $k = 1,\ldots,N$, to be free to vary in order to obtain a quadrature formula

$$\int_a^b f(x)dx \approx \sum_{k=1}^{N} A_k f(\nu_k)$$

of as high degree as possible. The highest degree formula which can be obtained has degree $2N-1$; it is the N-point Gauss-Legendre formula. Analogous formulas exist for arbitrary $[a,b]$ and $w(x)$.

The well known Gauss formulas have special names. These are listed in Table 1. The Gauss-Legendre and both types of Gauss-Chebyshev formulas are special cases of the general Gauss-Jacobi. The points ν_k, $k = 1,\ldots,N$, in a Gauss formula are the zeros of the orthogonal polynomial $P_N(x)$ corresponding to $[a,b]$ and $w(x)$; this is the polynomial which satisfies

$$\int_a^b w(x)P_N(x)Q_{N-1}(x)dx = 0$$

Table 1. Names of the well known Gauss quadrature formulas.

Interval [a, b]	Weight function w(x)	Name of the quadrature formula
[-1, 1]	1	Gauss-Legendre
[-1, 1]	$(1-x^2)^{-1/2}$	Gauss-Chebyshev of the first kind
[-1, 1]	$(1-x^2)^{1/2}$	Gauss-Chebyshev of the second kind
[-1, 1]	$(1-x)^{\alpha}(1+x)^{\beta}$ $\alpha > -1$, $\beta > -1$	Gauss-Jacobi
$[0, \infty)$	e^{-x}	Gauss-Legendre
$[0, \infty)$	$x^{\alpha}e^{-x}$ $\alpha > -1$	Generalized Gauss-Legendre
$(-\infty, \infty)$	e^{-x^2}	Gauss-Hermite

for all polynomials $Q_{N-1}(x)$ of degree $\leq N-1$. (See, e.g. [1; p. 23].) Extensive tables of the ν_k, A_k in the Gauss formulas listed in Table 1 are given by Stroud and Secrest [4]. The main disadvantage of these formulas is the fact that the ν_k, A_k, are, in general, irrational numbers which usually must be taken from tables.

A linear change of variable can be applied to any quadrature formula. For example, the interval [-1,1] can be transformed by a linear change of variable onto any finite length interval. Therefore a Gauss-Legendre formula can be used on any finite length interval. In general, the linear transformation which relates the interval $a \leq x \leq b$ to the interval $c \leq z \leq d$ is

$$x = \gamma z + \delta \qquad z = \frac{1}{\gamma}(x-\delta)$$

$$dx = \gamma dz \qquad \gamma = \frac{b-a}{d-c} \qquad \delta = \frac{ad-bc}{d-c}$$

Under this transformation a given formula

$$\int_a^b w(x)f(x)dx = \sum_{k=1}^{N} A_k f(x_k)$$

is transformed into

$$\int_c^d w^*(z)g(z)dz = \sum_{k=1}^{N} A_k^* g(z_k)$$

where $w^*(z) \equiv w(\gamma z + \delta)$ and

$$z_k \equiv \frac{1}{\gamma}(x_k - \delta) \qquad A_k^* \equiv \frac{1}{\gamma} A_k$$

$$k = 1,\ldots,N.$$

Now we consider integrals in which $w(x)$ is one of the univariate pdf's listed in Table 2. The basic properties of these pdf's are derived by Zellner [5; Appendix A]. Here $\Gamma(u)$ is the gamma function and $B(u,v) = \Gamma(u)\Gamma(v)/\Gamma(u+v)$ the beta function.

(i) The univariate normal pdf. We assume that f depends on the same parameters as does p. If we make the change of variable

$$z = \frac{x-\theta}{\sigma\sqrt{2}} \qquad dz = \frac{dx}{\sigma\sqrt{2}} \qquad x = \sigma\sqrt{2}\, z + \theta$$

then

$$\int_{-\infty}^{\infty} p(x|\theta, \sigma)f(x|\theta, \sigma)dx =$$

$$= \frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty} e^{-z^2} f(\sigma\sqrt{2}\, z + \theta|\theta, \sigma)dz$$

$$\approx \frac{1}{\sqrt{\pi}} \sum_{k=1}^{N} A_k f(\sigma\sqrt{2}\, \nu_k + \theta|\theta, \sigma) \qquad (5)$$

where ν_k, A_k, $k = 1,\ldots,N$, is the N-point Gauss-Hermite formula. In Appendix A below we give a computer subroutine for approximation (5) which uses one of the values $N = 4, 8, 12, 16, 20$. The function $f(x|\theta, \sigma)$ must be given by a Fortran FUNCTION subprogram. The program of Appendix A computes approximation (5) for $\theta = 2$, $\sigma = 1$ for the two functions

Table 2. Some Univariate Probability Density Functions

Function	Interval	Name
$p(x\|\theta, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp[-\frac{1}{2\sigma^2}(x-\theta)^2]$ $-\infty < \theta < \infty \quad 0 < \sigma < \infty$	$-\infty < x < \infty$	univariate normal
$p(x\|\theta, h, \nu) = [B(\frac{1}{2}, \frac{\nu}{2})]^{-1}(\frac{h}{\nu})^{1/2}[1 + \frac{h}{\nu}(x-\theta)^2]^{-(\nu+1)/2}$ $-\infty < \theta < \infty \quad 0 < h < \infty \quad 0 < \nu$	$-\infty < x < \infty$	univariate student t
$p(x\|\gamma, a) = \frac{1}{\Gamma(a)\gamma^a} x^{a-1} e^{-x/\gamma}$ $0 < a \quad 0 < \gamma$	$0 \le x < \infty$	gamma
$p(y\|\gamma, a) = \frac{2}{\Gamma(a)\gamma^a y^{2a+1}} e^{-1/\gamma y^2}$ $0 < a \quad 0 < \gamma$	$0 < y < \infty$	inverted gamma
$p(x\|a, b, c) = \frac{1}{cB(a, b)}(\frac{x}{c})^{a-1}(1 - \frac{x}{c})^{b-1}$	$0 \le x \le c$	beta

$$f_1 = \sqrt{x^2 + \theta^2 + 1} \qquad f_2 = \frac{x^4 + 16}{x^2 + 1} \quad . \qquad (6)$$

The numerical results are given in Table 3. In each case the true value of the integral is not known. However, from the sequence of approximations obtained, the result for $N = 20$ appears to be accurate to at least 9 significant figures for f_1 and to between 3 and 4 figures for f_2.

(ii) The univariate student t pdf. To evaluate

$$\int_{-\infty}^{\infty} p(x|\theta, h, \nu) f(x) dx$$

we write this as the sum of two integrals

$$K\int_{-\infty}^{\theta} [1 + \frac{h}{\nu}(x-\theta)^2]^{-(\nu+1)/2} f_-(x)dx + K\int_{\theta}^{\infty} [1 + \frac{h}{\nu}(x-\theta)^2]^{-(\nu+1)/2} f_+(x)dx \qquad (7)$$

Table 3. Examples of approximation (5) for functions (6)

N	f_1	f_2
4	3.099350813737	8.636364
8	3.099454764011	8.882131
12	3.099453486117	8.835482
16	3.099453574338	8.845100
20	3.099453569961	8.846100

where

$$f(x) \equiv \begin{cases} f_-(x) & \text{for } x \le \theta \\ f_+(x) & \text{for } \theta \le x \end{cases}$$

and $K = [B(\frac{1}{2}, \frac{\nu}{2})]^{-1}(\frac{h}{\nu})^{1/2}$. Here f may also depend on θ, h, ν, but we will not indicate this with our notation. In (7) we make the change of variable

$$z = \frac{1}{1 + \frac{h}{\nu}(x-\theta)^2} \qquad x = \theta \pm \sqrt{\frac{\nu(1-z)}{hz}}$$

$$dx = \mp \frac{1}{2}\sqrt{\frac{\nu}{h}}\, z^{-3/2}(1-z)^{-1/2}\, dz \tag{8}$$

where in (8) the upper sign is used with f_+ and the lower sign with f_-. Then (7) becomes

$$\frac{K}{2}\sqrt{\frac{\nu}{2}} \int_0^1 z^{\nu/2-1}\,(1-z)^{-1/2}\, [f_-(\theta - \sqrt{\frac{\nu(1-z)}{hz}}\,) + f_+(\theta + \sqrt{\frac{\nu(1-z)}{hz}}\,)]dz \tag{9}$$

The weight function in (9), $w(z) = z^{\nu/2-1}(1-z)^{-1/2}$ is a Gauss-Jacobi weight function for [0,1]. Three important special cases are $\nu = 1, 2, 3$. For $\nu = 1$ the quadrature formulas are the Gauss-Chebyshev of the first kind; these are known in closed form:

$$\int_0^1 z^{-1/2}(1-z)^{-1/2}\, g(z)dz \simeq \sum_{k=1}^{N} A_k\, g(\nu_k) \tag{10}$$

$$A_k = \frac{\pi}{N} \qquad \nu_k = \frac{1}{2}\,[1 + \cos\frac{(2k-1)\pi}{2N}\,] \quad k = 1,\ldots,N$$

For $\nu = 3$ the formulas are also known in closed form:

$$\int_0^1 z^{1/2}(1-z)^{-1/2}\, g(z)dz \simeq \sum_{k=1}^{N} A_k g(\nu_k) \tag{11}$$

$$A_k = \frac{2\pi}{2N+1}\,\nu_k \qquad \nu_k = [\cos\frac{(2k-1)\,\pi}{2(2N+1)}]^2 \quad k = 1,\ldots,N$$

For $\nu = 2$ the A_k, ν_k in

$$\int_0^1 (1-z)^{-1/2}\, g(z)dz \simeq \sum_{k=1}^{N} A_k\, g(\nu_k) \tag{12}$$

are easily found from the (2N+1)-point Gauss-Legendre formula. (See [1; p. 143-144] for the details of (12) and also for (10), (11).) Stroud and Secrest [4; p. 29] give computer subroutines for computing Gauss-Jacobi formulas for general α and β.

(iii) The gamma pdf. To evaluate

$$\int_0^\infty p(x\,|\gamma,\, a)f(x)dx$$

we make the change of variable

$$z = x/\gamma \qquad x = \gamma z \qquad dx = \gamma dz\,.$$

Then the integral becomes

$$\frac{1}{\Gamma(a)} \int_0^\infty z^{a-1}\, e^{-z}\, f(\gamma z)dz \tag{13}$$

which can be evaluated by a generalized Gauss-Legendre formula. Stroud and Secrest [4] give computer subroutines for computing formulas for (13) and extensive tables for $a = 1$.

(iv) Inverted gamma pdf. To evaluate

$$\int_0^\infty p(y\,|\gamma,\, a)f(y)dy$$

we make the change of variable

$$z = \frac{1}{\gamma y^2} \qquad \frac{1}{y} = \sqrt{\gamma z} \qquad dy = -\frac{1}{2\gamma^{1/2}\, z^{3/2}}\, dz$$

Then the integral becomes

$$\frac{1}{\Gamma(a)} \int_0^\infty z^{a-1}\, e^{-z}\, f(\frac{1}{\sqrt{\gamma z}})dz$$

which can be approximated in the same way as (13).

(v) The beta pdf. This pdf is a constant

times the Gauss-Jacobi weight function for [0,c]. Therefore an integral

$$\int_0^c p(x|a, b, c)f(x)dx$$

can be approximated by Gauss-Jacobi formulas.

3. Integration Formulas for More Than One Variable

We discuss some approximations (2) which can be obtained by combinations of formulas for one variable. If we can find a transformation (usually nonlinear)

$$x_1 = x_1(u_1,\ldots,u_n)$$
$$\ldots\ldots$$
$$x_n = x_n(u_1,\ldots,u_n)$$

which transforms the integral of a monomial

$$\int\cdots\int_{R_n} w(x_1,\ldots,x_n)x_1^{\gamma_1}x_2^{\gamma_2}\cdots x_n^{\gamma_n}\,dx_1\cdots dx_n,$$

where the γ_i are nonnegative integers, into a product of n single integrals

$$\int_{a_k}^{b_k} w_k(u_k)Q_\gamma(u_k)du_k\ ,\ k = 1,\ldots,n$$

where $Q_\gamma(u_k)$ is a polynomial of degree $\gamma \le \gamma_1 + \gamma_2 + \ldots + \gamma_n$, then we can combine quadrature formulas for these single integrals to give a formula for R_n.

The most important regions for which such product formulas can be obtained are listed in Table 4. For the last four regions listed in this table the weight function can be more general than the one listed; we will not be interested in considering the most general case. Product formulas for other regions are considered in [3].

We will summarize product formulas for C_n, T_n, S_n and $E_n^{r^2}$. Product formulas for U_n are similar to those for S_n. Of course, one can apply a linear transformation to any formula. Thus, a formula for C_n can be used for any n-dimensional parallelogram; a formula for T_n can be used for any n-dimensional simplex; and a formula for S_n can be used for any n-dimensional ellipsoid.

Table 4. Some regions and weight functions for which product formulas can be obtained.

$$r^2 = x_1^2 + x_2^2 + \ldots + x_n^2$$

region name	region symbol	region definition	weight function
n-cube	C_n	$-1 \le x_k \le 1$, $k=1,\ldots,n$	$\prod_{k=1}^{n} w_k(x_k)$
n-simplex	T_n	$x_1 + x_2 + \ldots + x_n \le 1$, $x_k \ge 0$, $k = 1,\ldots,n$	1
n-sphere	S_n	$r^2 \le 1$	1
surface of n-sphere	U_n	$r^2 = 1$	1
entire n-space	$E_n^{r^2}$	$-\infty < x_k < \infty$, $k = 1,\ldots,n$	e^{-r^2}

For the n-cube C_n assume

$$w(x_1,\ldots,x_n) = w_1(x_1)w_2(x_2)\ldots w_n(x_n).$$

Then

$$\int_{-1}^{1}\cdots\int_{-1}^{1} w(x_1,\ldots,x_n)x_1^{\gamma_1}x_2^{\gamma_2}\ldots x_n^{\gamma_n}dx_1\ldots dx_n$$

is the product of

$$\int_{-1}^{1} w_k(x_k)x_k^{\gamma_k}dx_k \qquad k = 1,\ldots,n.$$

Therefore no transformation is required. If, for each k,

$$w_k(x_k) = (1 - x_k)^{\alpha_k}(1 + x_k)^{\beta_k},\ \alpha_k > -1,\ \beta_k > -1$$

then

$$\int_{-1}^{1}\cdots\int_{-1}^{1} w_1(x_1)\cdots w_n(x_n) f(x_1,\ldots,x_n)dx_1\cdots dx_n$$
$$\approx \sum_{j_1=1}^{M}\sum_{j_2=1}^{M}\cdots\sum_{j_n=1}^{M} A_{j_1}A_{j_2}\cdots A_{j_n} f(\nu_{j_1},\nu_{j_2},\ldots,\nu_{j_n}) \tag{14}$$

where, for each k,

$$A_{j_k},\ \nu_{j_k},\quad j_k = 1,\ldots,M$$

is a Gauss-Jacobi formula corresonding to $w_k(x_k)$. Approximation (14) has degree $2M-1$; it is called a Cartesian product formula to distinguish it from other product formulas.

Cartesian product formulas can also be obtained for $E_n^{r^2}$. Since the integral over entire n-space of $\exp(-r^2)$ times a monomial is a product of single integrals we can write

$$\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} e^{-x_1^2-\cdots-x_n^2} f(x_1,\ldots,x_n)dx_1\cdots dx_n$$
$$\approx \sum_{j_1=1}^{M}\sum_{j_2=1}^{M}\cdots\sum_{j_n=1}^{M} A_{j_1}A_{j_2}\cdots A_{j_n} f(\nu_{j_1},\nu_{j_2},\ldots,\nu_{j_n})$$

where, for each k, A_{j_k}, ν_{j_k}, $j_k = 1,\ldots,M$, is the M-point Gauss-Hermite formula. Below we will mention another type of product formula for $E_n^{r^2}$.

For the n-simplex T_n we transform an integral

$$\int_0^1\int_0^{1-x_1}\int_0^{1-x_1-x_2}\cdots\int_0^{1-x_1-\cdots-x_{n-1}} x_1^{\gamma_1}x_2^{\gamma_2}\cdots x_n^{\gamma_n}dx_n\cdots dx_1 \tag{15}$$

by making the change of variables

$$\begin{aligned} x_1 &= y_1 && = y_1\\ x_2 &= y_2(1-y_1) && = y_2(1-x_1)\\ x_3 &= y_3(1-y_2)(1-y_1) && = y_3(1-x_1-x_2)\\ &\cdots\\ x_n &= y_n(1-y_{n-1})\cdots(1-y_1) && = y_n(1-x_1-\cdots-x_{n-1}) \end{aligned} \tag{16}$$

Since the limits of integration for the x_k are

$$0 \le x_k \le 1 - x_1 - \cdots - x_{k-1},\quad k = 1,\ldots,n$$

the limits for the y_k will be

$$0 \le y_k \le 1,\quad k = 1,\ldots,n$$

The Jacobian of transformation (16) is

$$J = (1-y_1)^{n-1}(1-y_2)^{n-2}\cdots(1-y_{n-1})$$

Therefore integral (15) transforms into the product of the single integrals

$$\int_0^1 (1-y_k)^{n-k} Q_\gamma(y_k)dy_k \qquad k = 1,\ldots,n$$

where $Q_\gamma(y_k) = y_k^{\gamma_k}(1-y_k)^{\gamma_{k+1}+\cdots+\gamma_n}$.

It follows that if we have n one variable formulas of degree d of the form

$$\int_0^1 (1-y_k)^{n-k} g(y_k)\,dy_k \approx \sum_{j=1}^{M} A_{k,j}\, g(\mu_{k,j})$$
$$k = 1,\ldots,n$$

these can be combined to give a formula of degree d for T_n:

$$\int\cdots\int_{T_n} f(x_1,\ldots,x_n)\,dx_n\cdots dx_1$$
$$\approx \sum_{j_1=1}^{M}\cdots\sum_{j_n=1}^{M} A_{1,j_1}\cdots A_{n,j_n} f(\nu_{j_1},\nu_{j_1j_2},\ldots,\nu_{j_1j_2\cdots j_n}) \tag{17}$$

where

$$\begin{aligned} \nu_{j_1} &= \mu_{1,j_1}\\ \nu_{j_1j_2} &= \mu_{2,j_2}(1-\mu_{1,j_1})\\ &\cdots\\ \nu_{j_1j_2\cdots j_n} &= \mu_{n,j_n}(1-\mu_{n-1,j_{n-1}})\cdots(1-\mu_{1,j_1}) \end{aligned}$$

If the one variable formulas are Gauss-Jacobi formulas then formula (17) has degree $2M-1$. Formula (17) is called a conical product formula.

For the n-sphere S_n we transform to n-dimensional spherical coordinates $\phi_1,\phi_2,\ldots,\phi_{n-1},r$ as follows:

$$x_1 = r \cos\phi_{n-1} \cos\phi_{n-2} \cdots \cos\phi_2 \cos\phi_1$$
$$x_2 = r \cos\phi_{n-1} \cos\phi_{n-2} \cdots \cos\phi_2 \sin\phi_1$$
$$x_3 = r \cos\phi_{n-1} \cos\phi_{n-2} \cdots \sin\phi_2$$
$$\cdots$$
$$x_{n-1} = r \cos\phi_{n-1} \sin\phi_{n-2}$$
$$x_n = r \sin\phi_{n-1}$$

Then a monomial integral

$$\int_{S_n} \cdots \int x_1^{\gamma_1} x_2^{\gamma_2} \cdots x_n^{\gamma_n} \, dx_1 \cdots dx_n$$

transforms into the product of the integrals

$$\int_{-1}^{1} |r|^{n-1} Q_\gamma(r) dr$$

$$\int_{-\pi/2}^{\pi/2} (\cos\phi_k)^{k-1} Q_\gamma(\cos\phi_k, \sin\phi_k) d\phi_k \qquad (18)$$

$$k = 1, \ldots, n-1$$

where

$$Q_\gamma(r) = r^{\gamma_1+\gamma_2+\cdots+\gamma_n}$$

$$Q_\gamma(\cos\phi_k, \sin\phi_k) = (\cos\phi_k)^{\gamma_1+\cdots+\gamma_k} (\sin\phi_k)^{k+1}$$

$$k = 1, \ldots, n-1$$

Now we transform the integrals (18) by

$$y_k = \sin\phi_k \qquad 1 - y_k^2 = (\cos\phi_k)^2$$

$$dy_k = (\cos\phi_k) d\phi_k \qquad d\phi_k = (1-y_k^2)^{-1/2} dy_k$$

It follows (we do not give the complete argument) that if we have n one variable formulas of degree d

$$\int_{-1}^{1} |r|^{n-1} g(r) dr \simeq \sum_{j=1}^{M} A_{r,j} \, g(r_j)$$
$$\int_{-1}^{1} (1-y_k^2)^{k/2-1} g(y_k) dy_k \simeq \sum_{j=1}^{M} A_{k,j} \, g(y_{k,j}) \qquad (19)$$
$$k = 1, \ldots, n-1$$

these can be combined to give a formula of degree d for S_n:

$$\int_{S_n} \cdots \int f(x_1, \ldots, x_n) dx_1 \cdots dx_n \simeq$$
$$\simeq \sum_i A_{1,i_1} \cdots A_{n-1,i_{n-1}} A_{r,i_n} f(\nu_{i,1}, \ldots, \nu_{i,n}) \qquad (20)$$

where the summation in (20) is over

$$i \equiv (i_1, \ldots, i_n), \quad 1 \le i_k \le M, \quad k = 1, \ldots, n$$

and

$$\nu_{i,1} = r_{i_n} (1-y_{n-1,i_{n-1}}^2)^{1/2} \cdots (1-y_{2,i_2}^2)^{1/2} (1-y_{1,i_1}^2)^{1/2}$$

$$\nu_{i,2} = r_{i_n} (1-y_{n-1,i_{n-1}}^2)^{1/2} \cdots (1-y_{2,i_2}^2)^{1/2} y_{1,i_1}$$

$$\cdots\cdots$$

$$\nu_{i,n-1} = r_{i_n} (1-y_{n-1,i_{n-1}}^2)^{1/2} y_{n-2,i_{n-2}}$$

$$\nu_{i,n} = r_{i_n} y_{n-1,i_{n-1}}$$

Formula (20) is called a spherical product formula. If the one variable formulas are Gauss formulas then formula (20) will have degree $2M-1$.

Spherical product formulas can also be obtained for $E_n^{r^2}$. Here an integral

$$\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} e^{-x_1^2-\cdots-x_n^2} f(x_1, \ldots, x_n) dx_1 \cdots dx_n$$

is approximated by a sum analogous to the sum in (20). The only change which must be made in the sum in (20) is to take the

$$A_{r,i_n}, \; r_{i_n} \qquad i = 1, \ldots, M$$

to be a formula of the type

$$\int_{-\infty}^{\infty} |r|^{n-1} e^{-r^2} g(r) dr \simeq \sum_{j=1}^{M} A_{r,j} \, g(r_j)$$

Now we consider two special multivariate pdf weight functions. The basic properties of these are derived by Zellner [5; Appendix B].

(i) The multivariate normal pdf. This function

is

$$p(x|\theta, V) = \frac{|\det V|^{1/2}}{(2\pi)^{m/2}} \exp\{-\frac{1}{2}(x-\theta)'V(x-\theta)\}$$

where $x' = (x_1, x_2, \ldots, x_n)$, $\theta' = (\theta_1, \theta_2, \ldots, \theta_n)$ and V is an n by n positive definite symmetric matrix. To evaluate the integral

$$\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p(x|\theta, V) f(x) dx_1 \ldots dx_n \qquad (21)$$

we change to new variables $z' = (z_1, \ldots, z_n)$ defined by

$$x - \theta = \sqrt{2}\, Cz$$

where C is an n by n nonsingular matrix which satisfies

$$C'VC = I$$

where C' is the transpose of C and I is the identity matrix. Then (21) transforms into

$$\pi^{-n/2} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} e^{-z_1^2 - \ldots - z_n^2} f(\theta + \sqrt{2}Cz) dz_1 \ldots dz_n \qquad (22)$$

which can be approximated by either a cartesian product formula or a spherical product formula.

In Appendix B below we give a computer subprogram which approximates ingegral (22) for $n = 2$ using a spherical product formula with N^2 points of degree $2N-1$. N can be one of the values $N = 4, 8, 12, 16, 20$. Here vector θ is stored in array T and matrix V in the 2 by 2 array V. This program was used to approximate integral (21) for the two functions

$$f_1 = [x_1^2 + 4x_2^2 + 16]^{-1}$$

$$f_2 = \exp(\cos(2x_1 - x_2))$$

In both cases

$$(\theta_1, \theta_2) = (1,2) \qquad V = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$$

The program in Appendix B does the calculation for f_1. The numerical results are given in Table 5. For $N = 20$ the results for f_1 appear to be accurate to at least 11 significant figures and for f_2 to about 4 significant figures.

(ii) The multivariate student t pdf. This pdf is

$$p(x|\theta, V, \nu, n) = \frac{\nu^{\nu/2}\Gamma(\frac{\nu+n}{2})|\det V|^{1/2}}{\pi^{n/2}\Gamma(\nu/2)} [\nu + (x-\theta)'V(x-\theta)]^{-(\nu+n)/2}$$

where $x' = (x_1, \ldots, x_n)$, $\theta' = (\theta_1, \ldots, \theta_n)$ and V is an n by n positive definite symmetric matrix. To approximate

$$\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p(x|\theta, V, \nu, n) f(x) dx_1 \ldots dx_n \qquad (23)$$

we first change to variables $z_1, \ldots, z_n$ defined by

$$x - \theta = Cz, \; z' = (z_1, \ldots, z_n)$$

where C is an n by n matrix which satisfies

$$C'VC = I$$

Then (23) transforms into

$$\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p(z|\nu, n) f(\theta + Cz) dz_1 \ldots dz_n \qquad (24)$$

$$p(z|\nu, n) = \frac{\nu^{\nu/2}\,\Gamma(\frac{\nu+n}{2})}{\pi^{n/2}\Gamma(\nu/2)} [\nu + z_1^2 + \ldots + z_n^2]^{-(\nu+n)/2}$$

Next we transform to spherical coordinates by

$$z_1 = r \cos\phi_{n-1} \cos\phi_{n-2} \cdots \cos\phi_2 \cos\phi_1$$

$$z_2 = r \cos\phi_{n-1} \cos\phi_{n-2} \cdots \cos\phi_2 \sin\phi_1$$

$$z_3 = r \cos\phi_{n-1} \cos\phi_{n-2} \cdots \sin\phi_2$$

$$\cdots$$

$$z_{n-1} = r \cos\phi_{n-1} \sin\phi_{n-2}$$

$$z_n = r \sin\phi_{n-1}$$

Then (24) becomes

Table 5. Approximations obtained with the program of Appendix B.

N	f_1	f_2
4	0.030571497642006	1.295379
8	0.030572860067111	1.423888
12	0.030572856321073	1.444658
16	0.030572856349807	1.450727
20	0.030572856350007	1.452256

$$\int\cdots\int p(r,\phi_1,\ldots,\phi_{n-1})f_*(r,\phi_1,\ldots,\phi_{n-1})\,dr\,d\phi_1\ldots d\phi_{n-1} \tag{25}$$

$$p(r,\phi_1,\ldots,\phi_{n-1}) = \frac{\nu^{\nu/2}\,\Gamma(\frac{\nu+n}{2})\,r^{n-1}(\cos\phi_{n-1})^{n-2}\ldots(\cos\phi_3)^2(\cos\phi_2)}{\pi^{n/2}\,\Gamma(\nu/2)[\nu+r^2]^{(\nu+n)/2}}$$

where

$$0 \le r < \infty$$
$$-\pi/2 \le \phi_1 \le 3\pi/2$$
$$-\pi/2 \le \phi_k \le \pi/2 \qquad k = 2,\ldots,n-1$$

and where $f_*(r,\phi_1,\ldots,\phi_{n-1})$ corresponds to $f(\theta + Cz)$.

Now in (25) we transform r by

$$w = [\nu + r^2]^{-1} \qquad r = \sqrt{\frac{1-\nu w}{w}} \qquad dr = \frac{-dw}{2w^{3/2}(1-\nu w)^{1/2}}$$

Then (25) becomes

$$\int\cdots\int p(w,\phi_1,\ldots,\phi_{n-1})f_{**}(w,\phi_1,\ldots,\phi_{n-1})\,dw\,d\phi_1\ldots d\phi_{n-1} \tag{26}$$

$$p(w,\phi_1,\ldots,\phi_{n-1}) = \frac{\nu^{\nu/2}\,\Gamma(\frac{\nu+n}{2})}{2\pi^{n/2}\,\Gamma(\frac{\nu}{2})}\,w^{\nu/2-1}(1-\nu w)^{n/2-1}(\cos\phi_{n-1})^{n-2}\ldots(\cos\phi_2)$$

$$f_{**}(w,\phi_1,\ldots,\phi_{n-1}) = f_*\left(\sqrt{\frac{1-\nu w}{w}},\phi_1,\ldots,\phi_{n-1}\right)$$

$$0 \le w \le 1/\nu$$

It follows that integral (26) can be approximated by a spherical product of formulas of the following types:

$$\int_0^{1/\nu} w^{\nu/2-1}(1-\nu w)^{n/2-1}\,g(w)\,dw \approx \sum_{j=1}^{M} A_{w,j}\,g(w_j) \tag{27}$$

$$\int_{-\pi/2}^{3\pi/2} g(\cos\phi_1,\sin\phi_1)\,d\phi_1 \approx \sum_{j=1}^{2M} A_{1,j}\,g(\cos\phi_{1,j},\sin\phi_{1,j}) \tag{28}$$

$$\int_{-1}^{1} (1-y_k^2)^{k/2-1}g(y_k)\,dy_k \approx \sum_{j=1}^{M} A_{k,j}\,g(y_{k,j}) \tag{29}$$

$$y_k = \sin\phi_k, \quad k = 2,\ldots,n-1$$

Formula (27) is a Gauss-Jacobi formula. A suitable formula (28) consists of $2M$ angles $\phi_{1,j}$, $j = 1,\ldots,2M$ equally spaced in $0 \le \phi \le 2\pi$ with all the $A_{1,j}$ equal to π/M. Formulas (29) are the same as in (19).

4. Remarks

Computer subroutines which approximate univariate integrals by Gauss quadrature formulas can be found in some computer subroutine packages. The IBM System/360 Scientific Subroutine Package, Version III, 1970, contains subroutines for Gauss-Legendre, Gauss-Hermite and Gauss-Laguerre formulas.

5. Acknowledgments

The calculations described above were carried out at the Data Processing Center of Texas A&M University in Fortran double precision on the AMDAHL 470 V/6 computer which is compatible with the IBM 360/65. The preparation of this paper was supported in part by NSF Grant MCS76-82888.

REFERENCES

1. P. J. David and P. Rabinowitz, Methods of Numerical Integration, Academic Press, 1975.

2. V. I. Krylov, Approximate Calculation of Integrals, Macmillan, 1962 (translated from Russian).

3. A. H. Stroud, Approximate Calculation of Multiple Integrals, Prentice-Hall, 1971.

4. A. H. Stroud and D. Secrest, Gaussian Quadrature Formulas, Prentice-Hall, 1966.

5. A. Zellner, An Introduction to Bayesian Inference in Econometrics, Wiley & Sons, 1971.

```
C
C                          APPENDIX A
C
C
C          A COMPUTER SUBPROGRAM FOR APPROXIMATING
C          A SINGLE INTEGRAL WITH  THE  UNIVARIATE
C          NORMAL PDF AS THE WEIGHT FUNCTION.
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      EXTERNAL  F1,F2
      DO 4 N=4,20,4
        T = 2.D0
        S = 1.D0
        Q1 = PDFN1(F1,T,S,N)
        Q2 = PDFN1(F2,T,S,N)
        PRINT 2, N,Q1,Q2
    2   FORMAT(10X,I4,2E25.16)
    4   CONTINUE
      STOP
      END
C
      FUNCTION F1(X,T,S)
      IMPLICIT REAL*8(A-H,O-Z)
      F1 = DSQRT(X*X + T*T + 1.D0)
      RETURN
      END
C
      FUNCTION F2(X,T,S)
      IMPLICIT REAL*8(A-H,O-Z)
      F2 = (X**4 + 16.D0)/(X*X + 1.D0)
      RETURN
      END
C
      FUNCTION PDFN1(FCN,T,S,N)
      IMPLICIT REAL*8(A-H,O-Z)
C               THIS APPROXIMATES THE INTEGRAL OF
C
C               K*EXP(-(X-T)**2/(2*S**2))*FCN(X,T,S)
C                   K = 1./SQRT(2.*PI*S**2)
C
C             OVER    -INFINITY .LT. X .LT. INFINITY
C             USING THE N-POINT GAUSS-HERMITE QUADRATURE
C             FORMULA.  THE ALLOWED VALUES OF  N  ARE
C                     N = 4,8,12,16,20
C
      DIMENSION  X(5,10),A(5,10)
      DATA (X(1,J),A(1,J),J=1,2) /
     1      .16506801238857850+1 ,  .81312835447245180-1 ,
     2      .52464762327529030+0 ,  .80491409000551280+0 /
      DATA (X(2,J),A(2,J),J=1,4) /
     1      .29306374202572440+1 ,  .19960407221136760-3 ,
     2      .19816567566958430+1 ,  .17077983007413480-1 ,
     3      .11571937124467800+1 ,  .20780232581489190+0 ,
     4      .38118699020732210+0 ,  .66114701255824130+0 /
```

```
      DATA (X(3,J),A(3,J),J=1,6) /
     1       .3897248978697 82D+1 ,   .2658551684356302D-6 ,
     2       .3020637025120890D+1 ,   .8573687043587859D-4 ,
     3       .2279507080501060D+1 ,   .3905390584629062D-2 ,
     4       .1597682635152605D+1 ,   .5160798561588393D-1 ,
     5       .9477883912401637D+0 ,   .2604923102641611D+0 ,
     6       .3142403762543591D+0 ,   .5701352362624796D+0 /
      DATA (X(4,J),A(4,J),J=1,8) /
     1       .4688738939305818D+1 ,   .2654807474011182D-9 ,
     2       .3869447904860123D+1 ,   .2320980844865211D-6 ,
     3       .3176999161979956D+1 ,   .2711860092537882D-4 ,
     4       .2546202157847481D+1 ,   .9322840086241805D-3 ,
     5       .1951787990916254D+1 ,   .1288031153550997D-1 ,
     6       .1380258539198881D+1 ,   .8381004139898583D-1 ,
     7       .8229514491446559D+0 ,   .2806474585285337D+0 ,
     8       .2734810461381525D+0 ,   .5079294790166137D+0 /
      DATA (X(5,J),A(5,J),J=1,5) /
     1       .5387480890011233D+1 ,   .2229393645534151D-12,
     2       .4603682449550744D+1 ,   .4399340992273181D-9 ,
     3       .3944764040115625D+1 ,   .1086069370769282D-6 ,
     4       .3347854567383216D+1 ,   .7802556478532064D-5 ,
     5       .2788806058428130D+1 ,   .2283386360163540D-3 /
      DATA (X(5,J),A(5,J),J=6,10)/
     1       .2254974002089276D+1 ,   .3243773342237862D-2 ,
     2       .1738537712116586D+1 ,   .2481052088746361D-1 ,
     3       .1234076215395323D+1 ,   .1090172060200233D+0 ,
     4       .7374737285453944D+0 ,   .2866755053628341D+0 ,
     5       .2453407083009012D+0 ,   .4622436696006101D+0 /
      DATA SRPI,SR2/ 1.772453850905516, 1.414213562373095/
      NX = N
      IF(N.GT.20) NX=20
      IF(N.LT. 4) NX= 4
      N4 = NX/4
      N2 = 2*N4
      SSR2 = S*SR2
      QSUM = 0.D0
      DO 2 K=1,N2
        Z = T + SSR2*X(N4,K)
        W = T - SSR2*X(N4,K)
    2   QSUM = QSUM + A(N4,K)*(FCN(Z,T,S)+FCN(W,T,S))
      PDFN1 = QSUM/SRPI
      RETURN
      END
```

```
C
C                          APPENDIX B
C
C
C          A COMPUTER SUBPROGRAM FOR APPROXIMATING
C          A DOUBLE INTEGRAL  WITH  THE  BIVARIATE
C          NORMAL PDF AS THE WEIGHT FUNCTION.
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      EXTERNAL  F1
      DIMENSION  T(2),V(2,2)
      T(1) = 1.D0
      T(2) = 2.D0
      V(1,1) = 2.D0
      V(2,2) = 3.D0
      V(1,2) = 1.D0
      V(2,1) = 1.D0
      DO 4 N=4,20,4
        Q1 = PDFN2(F1,T,V,N)
        PRINT 2, N,Q1
    2   FORMAT(10X,I4,2E25.16)
    4   CONTINUE
      STOP
      END
C
      FUNCTION F1(X1,X2)
      IMPLICIT REAL*8(A-H,O-Z)
      F1 = 1.D0/(X1**2+4.D0*X2**2+16.D0)
      RETURN
      END
C
      FUNCTION PDFN2(FCN,T,V,N)
      IMPLICIT REAL*8(A-H,O-Z)
C                THIS APPROXIMATES THE INTEGRAL OF
C
C              K*EXP(-.5*(X-T)'*V*(X-T))*FCN(X1,X2)
C                X' = (X1,X2),    T' = (T(1),T(2))
C                V  = POS. DEF. SYM. 2 BY 2 MATRIX
C                K  = SQRT(DET(V))/(2.*PI)
C
C              OVER  -INFINITY .LT. X1,X2 .LT. INFINITY
C              USING THE N*N-POINT SPHERICAL PRODUCT
C              GAUSS FORMULA.  ALLOWED VALUES OF N ARE
C                       N = 4,8,12,16,20
C
      DIMENSION  R(5,10),B(5,10),T(2),V(2,2),C(2,2)
      DATA (R(1,J),B(1,J),J=1,2) /
     1       .765366864730179 5D+0 ,   .4267766952966369D+0 ,
     2       .1847759065022574D+1 ,   .7322330470336312D-1 /
      DATA (R(2,J),B(2,J),J=1,4) /
     1       .5679328213965031D+0 ,   .3015770521708168D+0 ,
     2       .1321272530993643D+1 ,   .1787093462188998D+0 ,
     3       .2129934340988268D+1 ,   .1944395425750269D-1 ,
     4       .3065137992375080D+1 ,   .2696473527806637D-3 /
```

```
      DATA (R(3,J),B(3,J),J=1,6) /
     1       .4720663133281814D+0 ,   .2294823697498180+0 ,
     2       .1090381631206535D+1 ,   .2085004153860605D+0 ,
     3       .1729952694746106D+1 ,   .5668691037022490-1 ,
     4       .2403152839314327D+1 ,   .5199598726574537D-2 ,
     5       .3136473723528158D+1 ,   .1305086014074660D-3 ,
     6       .3997858674415806D+1 ,   .4492739532148106D-6 /
      DATA (R(4,J),B(4,J),J=1,8) /
     1       .4126495272081394D+0 ,   .1845942946708188D+0 ,
     2       .9506323036797034D+0 ,   .2093933904071715D+0 ,
     3       .1500362166233917D+1 ,   .8789749331858590D-1 ,
     4       .2065599227896752D+1 ,   .1667174613060783D-1 ,
     5       .2654412440144422D+1 ,   .1397268117612836D-2 ,
     6       .3280017684431137D+1 ,   .4538254386679107D-4 ,
     7       .3967452411973961D+1 ,   .4242873358136266D-6 ,
     8       .4781540728352030D+1 ,   .5240005874357552D-9 /
      DATA (R(5,J),B(5,J),J=1,5) /
     1       .3712054290288498D+0 ,   .1542205578825101D+0 ,
     2       .8540811141239282D+0 ,   .2005599645776368D+0 ,
     3       .1344746407967062D+1 ,   .1090341438059047D+0 ,
     4       .1844297616398964D+1 ,   .3104372804933887D-1 ,
     5       .2356373514548108D+1 ,   .4750758487590550D-2 /
      DATA (R(5,J),B(5,J),J=6,10)/
     1       .2886200399619627D+1 ,   .3765041942937694D-3 ,
     2       .3441480181244702D+1 ,   .1412961674799783D-4 ,
     3       .4034756229486250D+1 ,   .2124656992481343D-6 ,
     4       .4690051792036071D+1 ,   .9197824119898154D-9 ,
     5       .5469981445331775D+1 ,   .4955913609804504D-12/
      DATA PI,SR2 / 3.141592653589793, 1.414213562373095 /
      C(1,1) = 1.D0/DSQRT(V(1,1))
      DETV   = V(1,1)*V(2,2)-V(1,2)**2
      C(2,2) = DSQRT(V(1,1)/DETV)
      C(1,2) = - C(2,2)*V(1,2)/V(1,1)
      C(2,1) = 0.D0
      NX = N
      IF(N.GT.20) NX=20
      IF(N.LT. 4) NX= 4
      N4 = NX/4
      N2 = 2*N4
      NX = 4*N4
      FN = NX
      PN = PI/FN
      QSUM = 0.D0
      DO 4 J=1,NX
        FJ = DFLOAT(J) - .5D0
        CS = DCOS(FJ*PN)
        SN = DSIN(FJ*PN)
        DO 2 K=1,N2
          W1 = SR2*R(N4,K)*CS
          W2 = SR2*R(N4,K)*SN
          X1 = T(1) + C(1,1)*W1 + C(1,2)*W2
          X2 = T(2) +             C(2,2)*W2
          U1 = T(1) - C(1,1)*W1 - C(1,2)*W2
          U2 = T(2) -             C(2,2)*W2
     2    QSUM = QSUM + B(N4,K)*(FCN(X1,X2)+FCN(U1,U2))
     4  CONTINUE
      PDFN2 = QSUM/FN
```

BAYESIAN ANALYSIS AND COMPUTING

by

Arnold Zellner*
University of Chicago

ABSTRACT

The paper provides computer scientists with an overview and some examples of computational problems associated with applications of standard Bayesian inference procedures. Bayesian concepts and operations are reviewed with emphasis placed on explaining the general computational needs of Bayesian analysts. To make the considerations concrete, features of Bayesian and non-Bayesian computer programs for the standard multiple regression model are compared. Several features of a Bayesian Regression Analysis Program (BRAP) recently developed at the U. of Chicago are described. Several computational problems associated with variants of the multiple regression model are discussed and a discussion of certain specific computational needs of Bayesian analysts is presented.

I. Introduction

The main purpose of this paper is to provide computer scientists with an overview and some examples of computational problems associated with applications of standard Bayesian inference procedures. Bayesian inference procedures are a unified set of procedures in the sense that the same principles are applied in analyses of a very broad range of statistical problems. Thus it is important that computer scientists working in the area of Bayesian computing be familiar with basic Bayesian principles, concepts and operations in order to facilitate their interaction with Bayesian analysts and to understand the nature of Bayesian analyses.

The plan of the paper is as follows. In Section II, some basic Bayesian concepts and operations are reviewed with emphasis placed on explaining the general computational needs of Bayesian analysts. To make the considerations in Section II concrete, Section III is devoted to a comparison of features of a Bayesian and non-Bayesian computer programs for the standard multiple regression model. Several features of a Bayesian Regression Analysis Program (BRAP) recently developed at the U. of Chicago are described. In Section IV, several computational problems associated with variants of the multiple regression model are discussed. Finally in Section V, concluding comments and a discussion of certain specific computational needs of Bayesian analysts are presented.

II. Overview of Some Standard Principles of Bayesian Inference

The principles presented below are treated in a number of Bayesian texts including Jeffreys (1939, 1948, 1961, 1967), Raiffa and Schlaifer (1961), Lindley (1965), DeGroot (1970), Zellner (1971) and Box and Tiao (1973).

2.1 Bayes' Theorem and Posterior Probability Density Functions for Parameters

Central in Bayesian analysis is Bayes' Theorem or Rule. Given a random $n\times 1$ observation vector, $\underset{\sim}{y}$, and a $k\times 1$ random parameter vector, $\underset{\sim}{\theta}$, with joint probability density function (pdf), $p(\underset{\sim}{\theta},\underset{\sim}{y})$, $\underset{\sim}{\theta}\in\Theta$ and $\underset{\sim}{y}\in R_{\underset{\sim}{y}}$, we can write $p(\underset{\sim}{\theta},\underset{\sim}{y}) = p(\underset{\sim}{\theta})p(\underset{\sim}{y}|\underset{\sim}{\theta}) = p(\underset{\sim}{y})p(\underset{\sim}{\theta}|\underset{\sim}{y})$ where $p(\underset{\sim}{\theta})$ and $p(\underset{\sim}{y})$ are marginal pdfs and $p(\underset{\sim}{y}|\underset{\sim}{\theta})$ and $p(\underset{\sim}{\theta}|\underset{\sim}{y})$ are conditional pdfs. Then Bayes' Theorem or Rule is given by

$$p(\underset{\sim}{\theta}|\underset{\sim}{y}) = p(\underset{\sim}{\theta})p(\underset{\sim}{y}|\underset{\sim}{\theta})/p(\underset{\sim}{y}) \propto p(\underset{\sim}{\theta})p(\underset{\sim}{y}|\underset{\sim}{\theta}) \qquad (2.1a)$$

In (2.1),

$p(\underset{\sim}{\theta}|\underset{\sim}{y})$ ≡ posterior pdf

$p(\underset{\sim}{\theta})$ ≡ prior pdf

$p(\underset{\sim}{y}|\underset{\sim}{\theta})$ ≡ pdf for observations given $\underset{\sim}{\theta}$ or likelihood function

$p(\underset{\sim}{y})$ ≡ marginal pdf for $\underset{\sim}{y} = \int_{\Theta} p(\underset{\sim}{\theta})p(\underset{\sim}{y}|\underset{\sim}{\theta})d\underset{\sim}{\theta}$

*Research financed in part by NSF Grant SOC 77-15662 and by income from the H.G.B. Alexander Endowment Fund, Graduate School of Business, U. of Chicago.

Verbally (2.1a) may be expressed as,

$$\text{Posterior pdf} \propto (\text{Prior pdf}) \times (\text{Likelihood function}) \quad (2.1b)$$

In applications of (2.1a-b), the Bayesian analyst supplies the prior pdf $p(\underline{\theta})$ and the likelihood function $p(\underline{y}|\underline{\theta})$. The analyst's interest is centered on the posterior pdf $p(\underline{\theta}|\underline{y})$ because it summarizes the information contained in the prior pdf and likelihood function regarding possible values of the parameter vector. The analyst will generally wish to perform calculations to analyze various properties of the joint posterior pdf $p(\underline{\theta}|\underline{y})$. For example, he may wish to determine the modal value (or values) of $p(\underline{\theta}|\underline{y})$. Usually analysts are interested in evaluating means and other moments associated with $p(\underline{\theta}|\underline{y})$ as given by the following integrals:

$$\bar{\theta}_i \equiv \text{Posterior mean of } \theta_i \equiv \int_{\underline{\theta}} \theta_i p(\underline{\theta}|\underline{y}) d\underline{\theta} \qquad i=1,2,\ldots,k \quad (2.2a)$$

$$\text{Posterior variance of } \theta_i \equiv \int_{\underline{\theta}} (\theta_i - \bar{\theta}_i)^2 p(\underline{\theta}|\underline{y}) d\underline{\theta} \quad (2.2b)$$

Also, on occasion, computation of higher moments, e.g. $\int_{\underline{\theta}} (\theta_i - \bar{\theta}_i)^3 p(\underline{\theta}|\underline{y}) d\underline{\theta}$, $\int_{\underline{\theta}} (\theta_i - \bar{\theta}_i)^4 p(\underline{\theta}|\underline{y}) d\underline{\theta}$, etc. and measures of skewness and kurtosis are of interest.

In addition to the quantities mentioned above, analysts usually wish to compute the marginal pdfs associated with $p(\underline{\theta}|\underline{y}) = p(\theta_1, \theta_2, \ldots, \theta_k|\underline{y})$, that is e.g. the marginal posterior pdf for θ_1 given by

$$p(\theta_1|\underline{y}) = \int_{\theta_o} p(\theta_1, \theta_2, \ldots, \theta_k|\underline{y}) d\theta_2 d\theta_3 \ldots d\theta_k \quad (2.3)$$

where θ_o is the sub-space for $\theta_2, \theta_3, \ldots, \theta_k$. Given that $p(\theta_1|\underline{y})$ has been obtained, integrals of the following kind are often of interest to Bayesian analysts:

$$p\{a < \bar{\theta}_1 < b\} = \int_a^b p(\theta_1|\underline{y}) d\theta_1 \quad (a,b \text{ given}) \quad (2.4)$$

The quantities in (2.2)-(2.4) are some of the items relating to the joint posterior pdf $p(\underline{\theta}|\underline{y})$ that Bayesian analysts will generally wish to compute. As is apparent, these quantities are just standard ones that characterize certain properties of multivariate pdfs. Also linear combinations of the elements of $\underline{\theta}$, say $\underline{\eta} = C\underline{\theta}$, where C is a given $q \times k$ matrix of rank q are of interest in many problems. There is thus a need to obtain the pdf for the $q \times 1$ vector $\underline{\eta}$ and measures (means, variances, etc.) to characterize its properties.

With regard to the calculations described above, it is useful to distinguish two cases:

(1) $p(\underline{\theta}|\underline{y})$ has a "standard" form, e.g. multivariate normal or multivariate Student-t, such that the integrals in (2.2)-(2.4), etc., can be evaluated analytically or by use of standard tables, and

(2) $p(\underline{\theta}|\underline{y})$ has a "non-standard" form for which the integrals in (2.2)-(2.4), etc., cannot be evaluated analytically or using tabled results. In case (1), the Bayesian analyst can provide explicit formulas for the quantities like those in (2.2)-(2.3) and tables will be available for computing integrals such as that shown in (2.4). On the other hand in Case (2), it is necessary to resort to approximations and/or numerical integration to evaluate quantities such as shown in (2.2)-(2.4). Some examples illustrating Cases (1) and (2) will be presented below.

2.2 Bayesian Predictive Pdfs

Assume that the posterior pdf, $p(\underline{\theta}|\underline{y})$ is available and that it is desired to obtain the pdf for a $q \times 1$ vector of as yet unobserved observations $\tilde{\underline{y}}_*$ with pdf $p(\underline{y}_*|\underline{\theta})$, $\underline{y}_* \in R_{\underline{y}_*}$. The predictive pdf for $\tilde{\underline{y}}_*$, denoted by $p(\underline{y}_*|\underline{y})$ is given by:

$$p(\underline{y}_*|\underline{y}) = \int_{\underline{\theta}} p(\underline{y}_*|\underline{\theta}) p(\underline{\theta}|\underline{y}) d\underline{\theta} \quad (2.5)$$

For many problems, the integration in (2.5) can be done analytically while for others it has to be performed numerically. Given that the predictive pdf $p(\underline{y}_*|\underline{y})$ has been obtained, Bayesian analysts are interested in obtaining computational results that characterize its properties. Just as with a posterior pdf for a parameter vector, $p(\underline{\theta}|\underline{y})$, quantities analogous to those in (2.2)-(2.4) will often be evaluated to provide predictive means, variances and other measures associated with the multivariate predictive pdf, $p(\underline{y}_*|\underline{y})$. In addition, interest is often centered on given linear combinations of the element of $\underline{y}_*$, e.g. $\eta = \sum_{i=1}^{q} c_i y_{*i}$ with the c_i's given. The pdf for η and measures (mean, variance, etc.) characterizing it are required in Bayesian analyses.

The two cases mentioned at the end of Section 2.1 are also relevant with respect to the predictive pdf $p(\underline{y}_*|\underline{y})$. It may have a "standard" form or it may be in a "non-standard" form. In the latter case approximations, e.g. asymptotic expansions and/or numerical integration procedures will be required.

2.3 Bayesian Posterior Odds Ratios

In comparing alternative hypotheses or models, Bayesians employ posterior odds ratios. Consider data vector $\underline{y}$ with pdf $p_1(\underline{y}|\underline{\theta}_1)$ under hypothesis (or model) H_1 with parameter vector $\underline{\theta}_1$ and with pdf $p_2(\underline{y}|\underline{\theta}_2)$ under hypothesis (or model) H_2. Given that H_1 and H_2 are mutually exclusive and that H_1 and H_2 have prior probabilities π_1 and π_2, respectively, Bayes' Theorem* provides the following expression for the posterior odds ratio, K_{12}, for H_1 relative, to H_2:

$$K_{12} = \frac{\pi_1}{\pi_2} \cdot \frac{\int_{\underline{\theta}_1} p_1(\underline{y}|\underline{\theta}_1) p_1(\underline{\theta}_1|\cdot) d\underline{\theta}_1}{\int_{\underline{\theta}_2} p_2(\underline{y}|\underline{\theta}_2) p_2(\underline{\theta}_2|\cdot) d\underline{\theta}_2} \quad (2.6a)$$

$$= (\text{Prior Odds}) \times (\text{Ratio of Averaged Likelihoods}) \quad (2.6b)$$

where π_1/π_2 = Prior Odds, $p_1(\underline{\theta}_1|\cdot)$ and $p_2(\underline{\theta}_2|\cdot)$ are the prior pdfs for $\underline{\theta}_1$ and $\underline{\theta}_2$, respectively, and θ_1 and θ_2 are the domains of $\underline{\theta}_1$ and $\underline{\theta}_2$, respectively. The Ratio

---

*Note that $p(\underline{y},H) = p(H|\underline{y})p(\underline{y}) = p(\underline{y}|H)p(H)$ where $H=H_i$ with probability π_i, $i=1,2$. Then $p(H|\underline{y}) = p(H)p(\underline{y}|H)/p(\underline{y})$ and $K_{12} = p(H_1|\underline{y})/p(H_2|\underline{y}) = [p(H_1)/p(H_2)] \times [p_1(\underline{y}|H_1)/p_2(\underline{y}|H_2)]$. Since $p_i(H_i) = \pi_i$ and $p_i(\underline{y}|H_i) = \int_{\underline{\theta}_i} p_i(\underline{y}|\underline{\theta}_i) p_i(\underline{\theta}_i|\cdot) d\underline{\theta}_i$, for $i=1,2$, K_{12} is given by the expression in (2.6a)

of Averaged Likelihoods, or Bayes' Factor, is the ratio of integrals in (2.6a).

In applications, users will provide π_1/π_2, $p_i(\underline{y}|\underline{\theta}_i)$, $p_i(\underline{\theta}_i|\cdot)$, and θ_i, i=1,2. In some problems the integrals involved in (2.6a) can be evaluated analytically while in others their values will have to be approximated either by asymptotic expansions or by numerical integration procedures.

The discussion above has considered some standard Bayesian operations, concepts and computational requirements. To make these considerations more concrete, in the next Section comparative aspects of Bayesian and non-Bayesian regression analyses computer programs will be discussed.

III. Bayesian and Non-Bayesian Regression Analysis Computer Programs

Since the standard multiple regression model (MRM) is a central one in many areas of science, specific attention is devoted to a summary of Bayesian and non-Bayesian analysis of it that features computational problems. The MRM for the $n\times 1$ observation vector $\underline{y}$ is given by

$$\underset{n\times 1}{\underline{y}} = \underset{n\times k}{X}\ \underset{k\times 1}{\underline{\beta}} + \underset{n\times 1}{\underline{u}} \qquad (3.1)$$

where X is an $n\times k$ nonstochastic matrix of rank k, $\underline{\beta}$ is a $k\times 1$ vector of regression coefficients with unknown values, and $\underline{u}$ is an $n\times 1$ error vector. It is assumed that $\underline{u}$ has been drawn from a multivariate normal distribution with zero mean vector and covariance matrix $\sigma^2 I_n$ where σ^2 is a parameter with unknown value. The likelihood function for the MRM is:

$$= (2\pi\sigma^2)^{-n/2}\exp\{-[\nu s^2 + (\underline{\beta}-\hat{\underline{\beta}})'X'X(\underline{\beta}-\hat{\underline{\beta}})]/2\sigma^2\}$$

where

$$\hat{\underline{\beta}} = (X'X)^{-1}X'\underline{y} \qquad (3.3a)$$

$$\nu s^2 = (\underline{y}-X\hat{\underline{\beta}})'(\underline{y}-X\hat{\underline{\beta}}) \qquad (3.3b)$$

and

$$\nu = n-k\ . \qquad (3.3c)$$

Since Bayesians place great importance on study of the likelihood function's properties, it is usual to compute $\hat{\underline{\beta}}$, s^2 and other measures to characterize the likelihood function's properties.*

Two prior pdfs that are often employed in analyzing the MRM are (1) a diffuse or "non-informative pdf that is appropriate when little outside information is available about possible values of $\underline{\beta}$ and σ and (2) a natural conjugate prior pdf that embodies outside information about parameters' values that is supplied by the Bayesian analyst.

3.1 Bayesian Analysis of the MRM with a Diffuse Prior Pdf

A diffuse prior pdf for $\underline{\beta}$ and σ is given by

$$p(\underline{\beta},\sigma) \propto 1/\sigma\ , \qquad (3.4)$$

that is the β_i's, and $\log\sigma$ are assumed uniformly and independently distributed over very large finite intervals. Alternatively, it is possible to regard (3.4) as defined for $-\infty < \beta_i < \infty$, $i = 1,2,\ldots,k$ and $0 < \sigma < \infty$ in which case (3.4) is an improper pdf.* In either case, multiplying (3.2) and (3.4) together provides the posterior pdf for the parameters $\underline{\beta}$ and σ, $p(\underline{\beta},\sigma|\underline{y},I_o)$, where I_o denotes diffuse prior information, namely

$$p(\underline{\beta},\sigma|\underline{y},I_o) = p(\underline{\beta},\sigma)p(\underline{y}|\underline{\beta},\sigma)$$

$$\propto \sigma^{-(n+1)}\exp\{-[\nu s^2+(\underline{\beta}-\hat{\underline{\beta}})'X'X(\underline{\beta}-\hat{\underline{\beta}})/2\sigma^2]\} \qquad (3.5)$$

in which the factor of proportionality is a normalizing constant. The joint posterior pdf for $\underline{\beta}$ and σ is one that can be analyzed analytically. That is, to obtain the joint marginal distribution of the elements of $\underline{\beta}$, (3.5) can be integrated analytically with respect to σ, $0 < \sigma < \infty$, to yield:

$$p(\underline{\beta}|\underline{y},I_o) = c(\nu + (\underline{\beta}-\hat{\underline{\beta}})'X'X(\underline{\beta}-\hat{\underline{\beta}})/s^2\}^{-(\nu+k)/2} \qquad (3.6)$$

where $c = \nu^{\nu/2}\Gamma(n/2)|V|^{1/2}/\pi^{k/2}\Gamma(\nu/2)$, with $V = X'X/s^2$ is the normalizing constant. The pdf in (3.6) is in the form of a multivariate-Student-t pdf --- see e.g. Raiffa and Schlaifer (1961), Zellner (1971) and Box and Tiao (1973) for it properties. In particular, its mean vector and covariance matrix are given by

$$E(\underline{\beta}|\underline{y},I_o) = \hat{\underline{\beta}} = (X'X)^{-1}X'\underline{y} \qquad \nu > 1 \quad (3.7)$$

and

$$E(\underline{\beta}-\hat{\underline{\beta}})(\underline{\beta}-\hat{\underline{\beta}})'|\underline{y},I_o) = (X'X)^{-1}\nu s^2/(\nu-2) \quad \nu > 2 \quad (3.8)$$

Further, from properties of (3.6), it is the case that the following quantity,

$$t_\nu = (\beta_i-\hat{\beta}_i)/s_{\beta_i}\ , \qquad (3.9)$$

where $\beta_i-\hat{\beta}_i$ is the i'th element of $\underline{\beta}-\hat{\underline{\beta}}$ and $s^2_{\beta_i} = m^{ii}/s^2$, where m^{ii} is the (i,i)th element of $(X'X)^{-1}$, has a univariate Student-t pdf with ν degrees of freedom.

In addition to the above well-known results, it is possible to obtain the posterior pdf for σ from (3.5) by integration with respect to the k elements of $\underline{\beta}$ to yield

$$p(\sigma|\underline{y},I_o) = c\sigma^{-(\nu+1)}\exp\{-\nu s^2/2\sigma^2\} \qquad (3.10)$$

with $c = 2(\nu s^2/2)^{\nu/2}/\Gamma(\nu/2)$, the normalizing constant. The posterior pdf for σ in (3.10) is in the form of an inverted gamma function --- see Raiffa and Schlaifer (1961), Zellner (1971) and Box and Tiao (1973) for its properties. From (3.10), it is possible to obtain the posterior pdfs for σ^2 and $h = 1/\sigma^2$ by simple changes of variable. Similarly, the quantity $\nu s^2/\sigma^2$ has a χ^2 pdf with ν degrees of freedom. Explicit algebraic expressions for means, medians, variances, etc., associated with posterior pdfs for σ, σ^2, and h are available.

Finally, the posterior pdf of a $q\times 1$ vector $\underline{\eta} = C\underline{\beta}$ where C is a $q\times k$ given matrix of rank q is in the form of a q-dimensional multivariate Student-t pdf with ν degrees of freedom, posterior mean vector

$$E(\underline{\eta}|\underline{y},I_o) = \hat{\underline{\eta}} = C\hat{\underline{\beta}} \qquad \nu > 1 \qquad (3.11)$$

---

*Here we have limited attention to a likelihood function based on an assumed normally distributed error vector. Other distributional assumptions can be made --- see e.g. Zellner (1976).

*Rényi (1970) provides an axiom system for probability theory that accomodates unbounded measures and a derivation of Bayes' Theorem within it.

and posterior covariance matrix

$$E[(\underline{\eta}-\hat{\underline{\eta}})(\underline{\eta}-\hat{\underline{\eta}})'|\underline{y},I_o] = CV(\underline{\beta}|\underline{y},I_o)C' \qquad \nu > 2 \quad (3.12)$$

where $V(\underline{\beta}|\underline{y},I_o) = (X'X)^{-1}\nu s^2/(\nu-2)$ is the posterior covariance matrix for $\underline{\beta}$. Each individual element of $\underline{\eta}$ has a posterior pdf in the form of a univariate Student-t pdf with ν degrees of freedom.

The results in (3.7)-(3.12) and others mentioned in the text are some of the items that are of interest to Bayesian analysts employing the MRM with the diffuse prior pdf in (3.4). It is seen that many quantities to be computed are similar to those computed in non-Bayesian least squares regression programs but have a fundamentally different interpretation. For example $\hat{\underline{\beta}} = (X'X)^{-1}X'\underline{y}$ in (3.7), the mean of the posterior pdf for $\underline{\beta}$, is numerically the same as the least squares (and maximum likelihood) estimate but has a fundamentally different interpretation as the mean of a posterior pdf that summarizes information regarding possible values of $\underline{\beta}$, a vector of parameters considered to be subjectively random. Further, the result in (3.9) can be employed to compute probabilities regarding β_i's value as follows. From the tables of the Student-t distribution with ν degrees of freedom, the value of a constant, $c_{\alpha/2}$, can be obtained such that $Pr\{-c_{\alpha/2} < t_\nu < c_{\alpha/2}|\underline{y},I_o\} = 1-\alpha$. By inserting the value of t_ν in (3.9) in this probability statement and rearranging terms, the result is:

$$Pr(\hat{\beta}_i - c_{\alpha/2}s_{\hat{\beta}_i} < \beta_i < \hat{\beta}_i + c_{\alpha/2}s_{\hat{\beta}_i}|\underline{y},I_o) = 1-\alpha \quad (3.13)$$

In words (3.13) reads, given $\underline{y}$ and diffuse prior information I_o, as shown in (3.4), the probability that β_i lies in the given interval $\hat{\beta}_i \pm c_{\alpha/2}s_{\hat{\beta}_i}$ is equal to $1-\alpha$. In (3.13) that defines a $1-\alpha$ level Bayesian Confidence or Credibility Interval, β_i is considered to be random and all other quantities given or non-random. In contrast a non-Bayesian sampling theory confidence interval for β_i is given by the following probability statement

$$Pr(\hat{\beta}_i - c_{\alpha/2}s_{\hat{\beta}_i} < \beta_i < \hat{\beta}_i + c_{\alpha/2}s_{\hat{\beta}_i}|\beta_i) = 1-\alpha \quad (3.14)$$

In (3.14) β_i is non-random while $\hat{\beta}_i$ and $s_{\hat{\beta}_i}$ are considered to be random. Thus (3.14) states that the probability that the random interval $\hat{\beta}_i \pm c_{\alpha/2}s_{\hat{\beta}_i}$ covers the given value β_i is $1-\alpha$. The non-Bayesian computes $\hat{\beta}_i \pm c_{\alpha/2}s_{\hat{\beta}_i}$ from his data and refers to (3.14) for its interpretation while the Bayesian computes the same interval and refers to (3.13) for its interpretation. In this problem the intervals are the same (not always the case) but the Bayesian and non-Bayesian interpretations are quite different.

3.2 Bayesian Analysis of the MRM with a Natural Conjugate Prior Pdf

The natural conjugate prior pdf for the parameters $\underline{\beta}$ and σ of the normal MRM, $p_C(\underline{\beta},\sigma|\cdot)$ is given by

$$p_C(\underline{\beta},\sigma|\cdot) = p_N(\underline{\beta}|\sigma,\cdot)p_{IG}(\sigma|\cdot) \quad (3.15a)$$

with

$$p_N(\underline{\beta}|\sigma,\underline{\bar{\beta}},A) \propto \sigma^{-k}\exp\{-(\underline{\beta}-\underline{\bar{\beta}})'A(\underline{\beta}-\underline{\bar{\beta}})/2\sigma^2\} \quad (3.15b)$$

and

$$p_{IG}(\sigma|\nu_o,s_o) \propto \sigma^{-(\nu_o+1)}\exp(-\nu_o s_o^2/2\sigma^2) \quad (3.15c)$$

In (3.15a) the natural conjugate prior is given as the product of a condition normal pdf for $\underline{\beta}$ given σ, $p_N(\underline{\beta}|\sigma,\cdot)$, times a marginal inverted gamma pdf for σ, $p_{IG}(\sigma|\cdot)$. Specific forms for these two pdfs are given in (3.15b,c) wherein the quantities $\underline{\bar{\beta}}$, A, ν_o and s_o^2 are parameters of the prior pdf whose values are to be supplied by the Bayesian analyst.

Users of Bayesian regression programs will generally be familiar with the natural conjugate prior pdf in (3.15). However, the problem of choosing values of the prior parameters $\underline{\bar{\beta}}$, A, ν_o and s_o so as to represent the available prior information well is not a trivial one. Thus the paper by Zellner (1972), that may be helpful in this regard was included in the User's Manual for BRAP, Abowd (1977). Further, on integrating (3.15a) over σ, $0 < \sigma < \infty$, the resulting marginal prior pdf for $\underline{\beta}$ is given by

$$p(\underline{\beta}|\underline{\bar{\beta}},A,\nu_o,s_o) \propto (\nu_o + (\underline{\beta}-\underline{\bar{\beta}})'A(\underline{\beta}-\underline{\bar{\beta}})/s_o^2)^{-(k+\nu_o)/2} \quad (3.16)$$

a pdf in the form of a k-dimensional multivariate-Student-t pdf with ν_o degrees of freedom. From (3.16), the prior mean vector, $E\underline{\beta}$ and prior covariance matrix for $\underline{\beta}$, $V(\underline{\beta})$, are

$$E\underline{\beta} = \underline{\bar{\beta}} \qquad \nu_o > 1 \quad (3.17)$$

and

$$V(\underline{\beta}) = E(\underline{\beta}-\underline{\bar{\beta}})(\underline{\beta}-\underline{\bar{\beta}})' = A^{-1}\nu_o s_o^2/(\nu_o-2) \qquad \nu_o > 2 \quad (3.18)$$

If $\nu_o = 1$, (3.16) becomes the multivariate Cauchy pdf with non-existent moments. In this case $\underline{\bar{\beta}}$ is to be interpreted as a location vector and $A^{-1}s_o^2$ as a dispersion matrix. From (3.18), it is apparent how values of ν_o and s_o affect the values of $V(\underline{\beta})$ for a particular choice of A. Thus users should be aware of the fact that choice of values of ν_o and s_o in the prior pdf for σ in (3.15c) has an impact on the prior pdf for $\underline{\beta}$ as shown in (3.18).

Given that values of the parameters $\underline{\bar{\beta}}$, A, s_o and ν_o in (3.15) have been assigned, the natural conjugate prior pdf can be multiplied by the likelihood function in (3.2) to obtain the joint posterior pdf for $\underline{\beta}$ and σ, namely

$$p(\underline{\beta},\sigma|\underline{y},I_c) \propto p_N(\underline{\beta}|\sigma,\cdot)p_{IG}(\sigma|\cdot)p(\underline{y}|\underline{\beta},\sigma) \quad (3.19)$$

where I_c denotes natural conjugate prior information. On substituting from (3.2) and (3.15) in (3.19) and completing the square on $\underline{\beta}$, the resulting posterior pdf is:

$$p(\underline{\beta},\sigma|\underline{y},I_c) \propto [\sigma^{-k}\exp(-(\underline{\beta}-\bar{\underline{\beta}})'M(\underline{\beta}-\bar{\underline{\beta}})/2\sigma^2)][\sigma^{-(\nu_1+1)}\exp(-\nu_1 s_1^2/2\sigma^2)] \quad (3.20)$$

where

$$M = A + X'X \quad (3.21a)$$

$$\bar{\underline{\beta}} = (A + X'X)^{-1}(A\underline{\bar{\beta}} + X'X\hat{\underline{\beta}}) \quad (3.21b)$$

$$\nu_1 = n-k+\nu_o = \nu+\nu_o \quad (3.21c)$$

$$\nu_1 s_1^2 = \nu s^2 + \nu_o s_o^2 + \hat{\underline{\beta}}'X'X\hat{\underline{\beta}} + \underline{\bar{\beta}}'A\underline{\bar{\beta}} - \bar{\underline{\beta}}'M\bar{\underline{\beta}} \quad (3.21d)$$

---

*If $\underline{\beta}$ and σ were independent in the prior, that is $p(\underline{\beta},\sigma) = p_1(\underline{\beta})p_2(\sigma)$, this effect will not be present. The dependence between $\underline{\beta}$ and σ in (3.15) can be thought of as arising from a prior assumption that $\underline{z} = \underline{\beta}/\sigma$ and σ are independent. If $p(\underline{z},\sigma) = p_N(\underline{z})p_{IG}(\sigma)$, with $p_N(\underline{z})$ a normal prior pdf for $\underline{z}$, then on transforming back to $\underline{\beta}$ and σ, their pdf will take the form $p(\underline{\beta},\sigma) = p_N(\underline{\beta}|\sigma)p_{IG}(\sigma)$.

From the form of (3.20), it is possible to integrate it analytically with respect to σ, $0 < \sigma < \infty$, to obtain the following marginal posterior pdf for $\underline{\beta}$,

$$p(\underline{\beta}|\underline{y},I_o) \propto (\nu_1 + (\underline{\beta}-\tilde{\underline{\beta}})'M(\underline{\beta}-\tilde{\underline{\beta}})/s_1^2)^{-(k+\nu_1)/2} \quad (3.22)$$

a pdf that is in the form of a k-dimensional multivariate Student-t pdf with $\nu_1 = \nu + \nu_o$ degrees of freedom with mean vector and covariance matrix given by

$$E(\underline{\beta}|\underline{y},I_o) = \tilde{\underline{\beta}} \quad (3.23)$$

and

$$V(\underline{\beta}|\underline{y},I_o) = M^{-1}\nu_1 s_1^2/(\nu_1-2) \quad (3.24)$$

Further, the quantity

$$t_{\nu_1} = (\beta_1-\tilde{\beta}_1)/s_{\tilde{\beta}_1} \quad (3.25)$$

has a univariate Student-t pdf with ν_1 degrees of freedom.

On comparing (3.6)-(3.9) with the corresponding equations (3.22)-(3.25), it is seen that they are in similar forms. While not shown here explicitly, the same similarity holds for the pdf for linear combinations of the elements of $\underline{\beta}$, $\underline{\eta} = C\underline{\beta}$ and for the predictive pdfs under the diffuse and natural conjugate priors. Thus it is convenient to program one multivariate Student-t routine

$$p(\underline{\theta}|\cdot) = c\{a + (\underline{\theta}-\hat{\underline{\theta}})'H(\underline{\theta}-\hat{\underline{\theta}})\}^{-(m+a)} \quad (3.26)$$

with $c = a^{a/2}\Gamma[(a+m)/2]|H|^{1/2}/\pi^{m/2}\Gamma(a/2)$

along with expressions for its moments, marginals, etc. For each particular case in which a multivariate Student-t pdf is required, appropriate values of a, the degrees of freedom parameter, m, the dimensionality, $\hat{\underline{\theta}}$, the location vector, and H the matrix of the quadratic form are inserted.

Also on integrating (3.20) with respect to the elements of $\underline{\beta}$, the result is

$$p(\sigma|\underline{y},I_o) = c\sigma^{-(\nu_1+1)}\exp\{-\nu_1 s_1^2/2\sigma^2\} \quad (3.27)$$

with $c = 2(\nu_1 s_1^2/2)^{\nu_1/2}/\Gamma(\nu_1/2)$. Note that (3.27) is in precisely the form of (3.10) and (3.15c). Thus it is useful to have a general "inverted-gamma pdf routine" in a Bayesian regression computer package that can be assigned appropriate parameter values for particular uses in diffuse and/or natural conjugate prior regression analyses.

Finally, just as with non-Bayesian regression programs it is useful to have goodness of fit measures and residuals computed in a Bayesian regression computer program. In Press and Zellner (1968), the posterior pdf for the population squared multiple correlation coefficient, ρ^2, is derived for the diffuse prior and natural conjugate prior cases. Also approximate large sample results are presented. In particular, for the diffuse prior case, it is shown that the posterior mean and variance of ρ^2 are given by,

$$E(\rho^2|\underline{y},I_o) \doteq R^2 \quad (3.28a)$$
$$\mathrm{Var}(\rho^2|\underline{y},I_o) \doteq 2R^2(2-R^2)(1-R^2)^2/n$$

where $R^2 = 1 - \hat{\underline{u}}'\hat{\underline{u}}/\sum_{i=1}^{n}(y_i-\bar{y})^2$, the squared sample multiple correlation coefficient, where $\hat{\underline{u}} = \underline{y} - X\hat{\underline{\beta}}$, the least squares residual vector, $\bar{y} = \sum_{i=1}^{n} y_i/n$, and n is the sample size.

As regards residual analysis or analysis of the realized error terms, Zellner (1975) has pointed out that the posterior mean of $\underline{u} = \underline{y} - X\underline{\beta}$ is given by

$$E(\underline{u}|\underline{y}) = \underline{y} - XE(\underline{\beta}|\underline{y}) \quad (3.29)$$

where $E(\underline{\beta}|\underline{y})$ is the posterior mean of $\underline{\beta}$. If a diffuse prior pdf is employed, $E(\underline{\beta}|\underline{y},I_o) = \hat{\underline{\beta}} = (X'X)^{-1}X'\underline{y}$ and thus $E(\underline{u}|\underline{y},I_o) = \underline{y} - X\hat{\underline{\beta}} = \hat{\underline{u}}$, the vector of least squares residuals. On the other hand, if a natural conjugate prior pdf is employed, the value for $E(\underline{\beta}|\underline{y})$ in (3.29) is given by (3.21b). These posterior means of $\underline{u}$ as well as other measures characterizing the properties of the realized error vector $\underline{u}$, mentioned in Zellner (1975), can be computed. This error term analysis provides information regarding the values of realized regression error terms.

From what has been presented above, it is clear that a Bayesian regression program incorporating use of diffuse and natural conjugate prior pdfs involves many calculations that are similar to those in non-Bayesian least squares programs. For example, good algorithms for inverting matrices such as $X'X$ and $X'X + A$ must be available. In addition, other algebraic operations are involved in computing posterior means and variances of regression coefficients, future observations, realized regression errors, Bayesian confidence and prediction intervals, etc. Finally, good plotting routines are useful in presenting results, for example comparative plots of marginal prior, posterior and predictive pdfs, contours of joint prior, posterior and predictive pdfs, posterior pdfs for realized error terms, etc. As with computer programs in general, it is important that the above calculations be carried out accurately and efficiently.

IV. Bayesian Analysis of Variants of the MRM

Many variants of the MRM have been analyzed from the Bayesian point of view --- see e.g. Zellner (1971, Ch. 4, 5 and 6), Box and Cox (1964), Tiao and Zellner (1964), Zellner and Tiao (1964), Zellner (1976), Richard (1977), Drèze (1977), etc. These analyses include treatments of autocorrelation, heteroscedasticity, transformations and other departures from assumptions of the standard MRM. Herein one aspect of the problem of heteroscedasticity will be reviewed to illustrate computational problems that arise in an analysis of a non-standard regression problem.

Assume that the data have been generated by the following system:

$$\begin{aligned} \underline{y}_1 &= X_1\underline{\beta} + \underline{u}_1 \\ \underline{y}_2 &= X_2\underline{\beta} + \underline{u}_2 \end{aligned} \quad (4.1)$$

where, for i=1,2, $\underline{y}_i$ is an $n_i \times 1$ vector, X_i is an $n_i \times k$ non-stochastic matrix of rank k, $\underline{\beta}$ is a $k \times 1$ regression coefficient vector and $\underline{u}_i$ is an $n_i \times 1$ vector of errors. Assume that the elements of $\underline{u}_1$ and $\underline{u}_2$ are normally and

independently distributed with zero means. The elements of $\underline{u}_1$ are assumed to have a common variance, σ_1^2, while those of $\underline{u}_2$ have a common variance, σ_2^2. Under these assumptions, the likelihood function is given by

$$p(\underline{Y}|\underline{\beta},\sigma_1,\sigma_2) \propto \sigma_1^{-n_1}\sigma_2^{-n_2}\exp(-(\underline{Y}_1-X_1\underline{\beta})'(\underline{Y}_1-X_1\underline{\beta})/2\sigma_1^2 - (\underline{Y}_2-X_2\underline{\beta})'(\underline{Y}_2-X_2\underline{\beta})/2\sigma_2^2) \quad (4.2)$$

where $\underline{Y}' = (\underline{Y}_1', \underline{Y}_2')$. Given that the following diffuse prior pdf for $\underline{\beta}$, σ_1 and σ_2 is employed, as in Tiao and Zellner (1964),

$$p(\underline{\beta},\sigma_1,\sigma_2) \propto 1/\sigma_1\sigma_2 , \quad (4.3)$$

the joint posterior pdf for the parameters is

$$p(\underline{\beta},\sigma_1,\sigma_2|\underline{Y},I_o) \propto \sigma^{-(n_1+1)}\sigma^{-(n_2+1)}\exp(-(\underline{Y}_1-X_1\underline{\beta})'(\underline{Y}_1-X_1\underline{\beta})/2\sigma_1^2-(\underline{Y}_2-X_2\underline{\beta})'(\underline{Y}_2-X_2\underline{\beta})/2\sigma_2^2) \quad (4.4)$$

The integration of (4.4) with respect to σ_1 and σ_2 $0 < \sigma_1, \sigma_2 < \infty$, can be done analytically to yield the following marginal posterior pdf for $\underline{\beta}$, namely

$$p(\underline{\beta}|\underline{Y},I_o) \propto [(\underline{Y}_1-X_1\underline{\beta})'(\underline{Y}_1-X_1\underline{\beta})]^{-n_1/2}[(\underline{Y}_2-X_2\underline{\beta}_2)'(\underline{Y}_2-X_2\underline{\beta})]^{-n_2/2} \propto [\nu_1 s_1^2 + (\underline{\beta}-\hat{\underline{\beta}}_1)'X_1'X_1(\underline{\beta}-\hat{\underline{\beta}}_1)]^{-n_1/2}[\nu_2 s_2^2 + (\beta-\hat{\underline{\beta}}_2)'X_2'X_2(\beta-\hat{\underline{\beta}}_2)]^{-n_1/2} \quad (4.5)$$

where, for $i=12$, $\nu_i s_i^2 = (\underline{Y}_i-X_i\hat{\underline{\beta}}_i)'(\underline{Y}_i-X_i\hat{\underline{\beta}}_i)$, $\hat{\underline{\beta}}_i = (X_i'X_i)^{-1}X_i'\underline{Y}_i$ and $\nu_i = n_i - k$. It is seen that (4.5) is in the form of a product of Student-t forms. Pdfs in this "double-t" form and in the form of more than two t-forms had been encountered earlier in Jeffreys' work (1961, p.140ff. and p.198ff.) relating to the analysis of data sets, with differing precisions, pertaining to measurement of a common mean. In connection with (4.5) that is a general form of Jeffreys' problem, Tiao and Zellner (1964) developed an asymptotic expansion approximation to the pdf and showed how moments of the pdf can be approximated.

After the asymptotic expansion approach for the analysis of (4.5) was put forward, it has been noted, apparently first by J. M. Dickey, that (4.4) can be expressed in terms of $\underline{\beta}$, σ_1 and $\lambda = \sigma_1^2/\sigma_2^2$. On making this change of variables, (4.4) becomes

$$p(\underline{\beta},\sigma_1,\lambda|\underline{Y},I_o) \propto \frac{\lambda^{(n_2-2)/2}}{\sigma_1^{n_1+n_2+1}}\exp\{-\frac{1}{2\sigma_1^2}[(\underline{Y}_1-X_1\underline{\beta})'(\underline{Y}_1-X_1\underline{\beta}) + \lambda(\underline{Y}_2-X_2\underline{\beta})'(\underline{Y}_2-X_2\underline{\beta})]\} \quad (4.6)$$

Now if we write,

$$\begin{bmatrix}\underline{Y}_1\\ \lambda^{\frac{1}{2}}\underline{Y}_2\end{bmatrix} = \begin{bmatrix}X_1\\ \lambda^{\frac{1}{2}}X_2\end{bmatrix}\underline{\beta} + \begin{bmatrix}\underline{u}_1\\ \lambda^{\frac{1}{2}}\underline{u}_2\end{bmatrix} \quad (4.7a)$$

or

$$\underline{w} = W\underline{\beta} + \underline{\varepsilon} \quad (4.7b)$$

where $\underline{w}' = (\underline{Y}_1', \lambda^{\frac{1}{2}}\underline{Y}_2')$, $W' = (X_1', \lambda^{\frac{1}{2}}X_2')$ and $\underline{\varepsilon} = (\underline{u}_1', \lambda^{\frac{1}{2}}\underline{u}_2')$, the quantity in square brackets in the exponential of (4.6) can be expressed as

$$(\underline{w}-W\underline{\beta})'(\underline{w}-W\underline{\beta}) = (\underline{w}-W\tilde{\underline{\beta}})'(\underline{w}-W\tilde{\underline{\beta}}) + (\underline{\beta}-\tilde{\underline{\beta}})'W'W(\underline{\beta}-\tilde{\underline{\beta}}) = \nu s^2 + (\underline{\beta}-\tilde{\underline{\beta}})'W'W(\underline{\beta}-\tilde{\underline{\beta}}) \quad (4.8)$$

where

$$\tilde{\underline{\beta}} = (W'W)^{-1}W'\underline{w} = (X_1'X_1 + \lambda X_2'X_2)^{-1}(X_1'\underline{Y}_1 + \lambda X_2'\underline{Y}_2) \quad (4.9)$$

$$\nu s^2 = (\underline{w}-W\tilde{\underline{\beta}})'(\underline{w}-W\tilde{\underline{\beta}}) \quad (4.10)$$

and $\nu = n_1 + n_2 - k$.

On inserting (4.8) in (4.4), the joint posterior pdf for $\underline{\beta}$, σ_1 and λ can be expressed as,

$$p(\underline{\beta},\sigma_1,\lambda|\underline{Y},I_o) \propto \frac{\lambda^{(n_2-2)/2}}{\sigma_1^{n_1+n_2+1}}\exp\{-\frac{1}{2\sigma_1^2}[\nu s^2 + (\underline{\beta}-\tilde{\underline{\beta}})'W'W(\underline{\beta}-\tilde{\underline{\beta}})]\} \quad (4.11)$$

wherein s^2, $\tilde{\underline{\beta}}$ and $W'W$ involve the parameter $\lambda = \sigma_1^2/\sigma_2^2$.

If (4.11) is integrated analytically with respect to σ_1, $0 < \sigma_1 < \infty$, and the elements of $\underline{\beta}$, $-\infty < \beta_i < \infty$, $i=1,2,\ldots,k$, the result is the marginal pdf for λ,

$$p(\lambda|\underline{Y},I_o) \propto \frac{\lambda^{(n_2-2)/2}}{(s^2)^{\nu/2}}|X_1'X_1 + \lambda X_2'X_2|^{-1/2} \quad 0 < \lambda < \infty \quad (4.12)$$

where $\nu = n_1 + n_2 - k$ and s^2, that depends on λ, is given in (4.10).. Univariate numerical integration techniques can be employed to obtain the normalizing constant and to analyze other features of the pdf in (4.12).

On integrating (4.11) analytically with respect to σ_1, $0 < \sigma_1 < \infty$, the result is the joint posterior pdf for $\underline{\beta}$ and λ, namely

$$p(\underline{\beta},\lambda|\underline{Y},I_o) \propto \lambda^{(n_2-2)/2}/\{\nu s^2 + (\underline{\beta}-\tilde{\underline{\beta}})'W'W(\underline{\beta}-\tilde{\underline{\beta}})\}^{(\nu+k)/2} \quad (4.13)$$

It is relevant to observe from (4.13) that the conditional posterior pdf for $\underline{\beta}$ given λ is in the following multivariate Student-t form:

$$p(\underline{\beta}|\lambda,\underline{Y},I_o) \propto \{\nu s^2 + (\underline{\beta}-\tilde{\underline{\beta}})'W'W(\underline{\beta}-\tilde{\underline{\beta}})\}^{-(\nu+k)/2}$$

and thus the conditional posterior pdf for any element of $\underline{\beta}$, say β_i, given λ will be in the univariate Student-t form; that is

$$\frac{\beta_i-\tilde{\beta}_i}{s_{\beta_i}} \quad \lambda \sim t_\nu \quad (4.14)$$

where $\tilde{\beta}_i$ is the i'th element of $\tilde{\underline{\beta}}$ and $s_{\beta_i}^2 = m^{ii}s^2$ where m^{ii} is the (i,i)th element of $(W'W)^{-1}$.

Thus given λ, the analysis of this problem is straightforward. Since λ can be estimated, that is $\hat{\lambda} = s_1^2/s_2^2$, it is possible to compute conditional posterior pdfs for the elements of $\underline{\beta}$ quite readily, given the conditioning on $\lambda = \hat{\lambda}$, and each of them will be in the univariate Student-t form as shown in (4.14). These are of course approximate results that will be reasonably accurate for moderate to large values of n_1 and n_2. Analyses can be performed by varying the conditionalizing value of λ to determine if results are sensitive to the value given to λ. If they are, it is important not to use the conditional posterior pdf for $\underline{\beta}$ given λ but rather to compute the exact marginal posterior pdfs for the elements of $\underline{\beta}$ from (4.13) as explained below.

To obtain the joint posterior pdf for a single element of $\underline{\beta}$, say β_i, and λ, we can employ

$$p(\beta_i,\lambda|\underline{y},I_o) = p(\beta_i|\lambda,\underline{y},I_o)p(\lambda|\underline{y},I_o),$$

with $p(\beta_i|\lambda,\underline{y},I_o)$ given by (4.14) and $p(\lambda|\underline{y},I_o)$ given in (4.12) to yield

$$p(\beta_i,\lambda|\underline{y},I_o) = \frac{\lambda^{(n_2-2)/2}}{s_{\beta_i}(s^2)^{\nu/2}} \frac{|X_1'X_1 + \lambda X_2'X_2|^{-1/2}}{\left[\nu + \left(\frac{\beta_i-\hat{\beta}_i}{s_{\beta_i}}\right)^2\right]^{\frac{\nu+1}{2}}} \qquad (4.15a)$$

or

$$p(\beta_i,\lambda|\underline{y},I_o) = \frac{\lambda^{(n_2-2)/2}}{\{m^{ii}|X_1'X_1 + \lambda X_2'X_2|\}^{1/2}} \frac{1}{[\nu s^2+(\beta_i-\hat{\beta}_i)^2/m^{ii}]^{(\nu+1)/2}} \qquad (4.15b)$$

where m^{ii} is the (i,i)'th element of $(X_1'X_1 + \lambda X_2'X_2)^{-1}$. The bivariate pdf in (4.15) requires application of bivariate numerical integration techniques to evaluate its normalizing constant, to obtain the marginal posterior pdf for β_i and to analyze other featuers of it.

The above problem is typical of many encountered in Bayesian econometrics and statistics in that a conditional analysis, for example conditioning on the value of λ above, leads to rather straightforward analytical results. Whether the conditional results are "accurate enough" is a moot point. If they are deemed not to be, perhaps because of sensitivity of final results to the conditioning, for example taking $\lambda = \hat{\lambda}$ above, then expressions like (4.15) have to be considered. Their analysis usually involves application of bivariate numerical integration techniques. It is essential that such techniques yield accurate results at as low a cost as possible. Note that in the problem analyzed above, numerical integration techniques will have to be applied in the analysis of (4.15) for $i=1,2,\ldots,k$, that is at least k bivariate integrals will have to be evaluated. This fact makes it essential to have an efficient and accurate bivariate numerical integration routine embedded in a MRM computer program.

V. Concluding Comments

From what has been presented above, it is seen that Bayesian analyses of the standard MRM, based on diffuse or natural conjugate prior distributions, involve many computations that are similar to those employed in usual least squares computer programs. These computations involve in the main matrix multiplication and matrix inversion. Thus usual accuracy and efficiency requirements for these operations are required in Bayesian regression programs. Further, in Bayesian regression computer programs, it will generally be necessary to have the capability of computing areas associated with standard pdfs --- e.g. univariate Student-t and inverted gamma pdfs, and of having such pdfs plotted as part of the output. Also, users will generally wish to have the capability of exploring the properties of alternative prior pdfs included in a Bayesian regression program. For the natural conjugate prior pdf, this involves analysis of Student-t, inverted gamma and associated gamma pdfs.

As regards "non-standard" regression problems, for example the heteroscedastic regression problem reviewed above, posterior distributions are usually not in forms that permit analytical results to be obtained and thus usually univariate and bivariate numerical integration routines must be employed. Also, if non-standard prior distributions are employed, for example a prior distribution that places finite ranges on the regression coefficients, multivariate numerical integration routines will generally be required in the analysis of posterior distributions. With respect to numerical integration routines, it is of course imperative that these be accurate and efficient. In this connection, there is the issue of the merits of Monte Carlo integration techniques relative to other techniques. Whatever technique is employed, it is desirable, if possible to have estimates of the errors in approximating integrals' values included in a program's output. Finally, in general it would be desirable to have information regarding the number of significant figures that can be guaranteed in the output of a program. For example, given that input data are accurate to three significant figures, it would be useful for the output of a program to contain statements regarding the number of significant figures that can be assured in the program's output.

In summary, it is the case that an operational Bayesian regression analysis computer program has many features in common with usual least squares regression programs. Thus many numerical analysis problems are common in both kinds of programs. However, for non-standard variants of the regression model, Bayesian regression computer programs require numerical integration procedures to get numerical approximations to exact finite sample posterior pdfs. On the other hand approximate Bayesian results, say based on conditional posterior pdfs, generally result in computations that are similar in many respects to the large sample approximate results, for example, approximate generalized least squares estimates, contained in non-Bayesian least squares computer programs.

REFERENCES

Abowd, J. (1977), "The Bayesian Regression Analysis Package: BRAP User's Manual, Version 1.0 of 9/8/77," H.G.B. Alexander Research Foundation, Graduate School of Business, U. of Chicago.

Box, G.E.P. and D. R. Cox (1964), "An Analysis of Transformations," J. Roy. Statist. Soc., Series B, 26, 211-243.

Box, G.E.P. and G. C. Tiao (1973), Bayesian Inference in Statistical Analysis, Reading, Mass.: Addison-Wesley Publishing Company.

DeGroot, M. H. (1970), Optimal Statistical Decisions, New York: McGraw-Hill Book Company.

Drèze, J. H. (1977), "Bayesian Regression Analysis Using Poly-t Densities," in A. Aykac and C. Brumat (editors), New Developments in the Applications of Bayesian Methods, Amsterdam: North-Holland Publishing Company, 153-184.

Jeffreys, H. (1939, 1948, 1961, 1967), Theory of Probability, Oxford: Oxford University Press.

Press, S. J. and A. Zellner (1968), "On the Posterior Distribution of the Squared Multiple Correlation Coefficient," H.G.B. Alexander Research Foundation, Graduate School of Business, U. of Chicago.

Raiffa, H. A. and R. S. Schlaifer (1961), Applied Statistical Decision Theory, Boston: Graduate School of Business Administration, Harvard U.

Rényi, A. (1970), Foundations of Probability, San Francisco: Holden-Day, Inc.

Richard, J. F. (1977), "Bayesian Analysis of the Regression Model When the Disturbances are Generated by an Auto-regressive Process," in A. Aykac and C. Brumat (editors), New Developments in the Applications of Bayesian Methods, Amsterdam: North-Holland Publishing Company, 185-209.

Tiao, G. C. and A. Zellner (1964), "Bayes' Theorem and the Use of Prior Knowledge in Regression Analysis," Biometrika, 51, 219-230.

Zellner, A. (1971), An Introduction to Bayesian Inference in Econometrics, New York: John Wiley and Sons, Inc.

_____ (1972), "On Assessing Informative Prior Distributions for Regression Coefficients," H.G.B. Alexander Research Foundation, Graduate School of Business, U. of Chicago.

_____ (1975), "Bayesian Analysis of Regression Errors," J. Am. Statist. Assoc., 70, No. 349, March, 138-144.

_____ (1976), "Bayesian and Non-Bayesian Analysis of the Regression Model with Multivariate Student-t Errors," J. Am. Statist. Assoc., 71, No. 354, June, 400-405.

Zellner, A. and G. C. Tiao (1964), "Bayesian Analysis of the Regression Model with Autocorrelated Errors," J. Am. Statist. Assoc., 59, No. 307, September, 763-778.

WORKSHOP 7

NUMERICAL ALGORITHMS

Chair: William J. Kennedy, Iowa State University

A COMPUTER PROGRAM FOR MODEL BUILDING WITH STEPWISE LOGISTIC REGRESSION

L. Engelman
University of California, Los Angeles
Health Sciences Computing Facility

Logistic regression is used to investigate the relation between a binary dependent variable and a set of independent variables. The dependent (outcome or response) variable represents events, such as success or failure, dead or alive, response or no response, etc. An independent (explanatory or covariate) variable may be categorical in nature, such as treatment, sex, race, etc., or an interval-scaled variable, such as age, temperature, heart rate, etc.

The program estimates the parameters (β_i) of the linear logistic model as the values that maximize the likelihood function; it proceeds in a stepwise manner, entering or removing one term from the model at each step. Selection of terms to be moved into or out of the model is based on either the likelihood ratio test or on a provisional asymptotic covariance matrix. The latter is considerably faster, making the computation of large problems practical. Interaction terms and design variable components are generated by the program for categorical variables. In the stepping process design variables are considered in sets, one set for each categorical variable or interaction term. Interactions are included in a hierarchical fashion; an interaction is not included if its lower order components are not included.

1. Introduction

In linear stepwise regression one tries to find a subset of independent variables that adequately predicts the dependent variable. One usually assumes that the dependent variable is continuous and has a constant (or at least a known) distribution.

In logistic regression one investigates the relation between a binary dependent variable and a set of independent variables. The dependent (outcome or response) variable $y = 1$ or 0 represents events such as dead or alive, success or failure, response or no response, etc. The outcome can be considered as a binomially distributed variable

$$\sum_{n} y = s \sim B(n,\theta) \qquad (1.1)$$

An independent (explanatory or covariate) variable may be categorical, such as treatment, sex, race, etc., or an interval-scaled variable, such as age, temperature, heart rate, etc.

The linear logistic model [3] is

$$E(\frac{s}{n}) = \theta = \frac{e^{\beta x}}{1+e^{\beta x}} \qquad (1.2)$$

where x represents the independent variables.

The maximum likelihood estimates of the parameters $(\beta_i) = \beta$ of (1.2) are obtained as the values $(\hat{\beta}_i)$ that maximize the function

$$L(\beta) = \frac{e^{y\beta' x}}{1+e^{\beta' x}} \qquad (1.3)$$

In stepwise logistic regression one also tries to find a subset of the independent variables that adequately predicts the dependent variable. This problem differs from the usual linear regression problem in that the estimates of β are nonlinear functions of the data that cannot be expressed in a closed form.

The computation proceeds in a stepwise manner. At each step a term is entered or removed from the model. Design variable components are generated for categorical variables and their interaction terms. In the stepping process design variables are considered in sets, one set for each categorical variable or interaction term. Interactions are included in a hierarchical fashion; an interaction is not included if its lower order components are not included.

Selection of a term to be moved into or out of the model at a step is based either on the likelihood ratio test (MLR) or on a provisional asymptotic covariance matrix estimate (ACE). While MLR is consistent with the estimation criterion, the ACE selection method is considerably faster, making the computations of large problems practical.

2. Other methods and models

The assumption of binomial distribution of s (1.1) is generally accepted. However, there have been at least four major suggestions as to the relation between θ and βx (see Cox [3] and Finney [6]):

- the logistic model (1.2)
- the linear model, which assumes that $d\theta/d(\beta x)$ has a uniform distribution in $0 \leq \beta x \leq 1$:

$$\theta = \lfloor \beta x \rceil_0^1 \ * \qquad (2.1)$$

- the cumulative normal model, which assumes that $d\theta/d(\beta x)$ has normal distribution:

$$\theta = \Phi(\beta x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\beta x} e^{-(\beta x)^2/2} d(\beta x) \qquad (2.2)$$

- the angular model, which assumes that $d\theta/d(\beta x)$ has angular distribution in $0 < \beta x \leq \pi/2$:

$$\theta = \sin^2(\lfloor \beta x \rceil_0^{\pi/2}) \qquad (2.3)$$

The two methods commonly used to estimate the parameters (β_i) are the maximum likelihood method and the minimum logit χ^2 method [1].

The maximum likelihood method estimates β as the value that maximizes

$$L(\beta) = \Pi f(\beta,x,y) = \Pi[y\ \theta(\beta x) + (1-y)(1 - \theta(\beta x))] \qquad (2.4)$$

The expression (1.3) is a specific case of (2.4).

The minimum χ^2 method estimates β as the value that minimizes

$$\chi^2 = np(1-p)[(t(p) - \beta x]^2 \qquad (2.5)$$

where t is the transformation that linearizes the model and $p = \hat{\theta}$, the observed proportion of $y = 1$. For the logistic distribution

$$t(\theta) = \log_e \frac{\theta}{1-\theta} = \beta x \qquad (2.6)$$

The expression (2.5) is the weighted least squares criterion to fit $\log_n(\frac{\theta}{1-\theta})$ if σ_θ^2 is assumed to be constant. Unfortunately, when $n = 1$, p is 0 or 1 and expression (2.5) becomes zero.

Table 1 describes the four models and the two methods.

Model \ Method	$t(\theta)$ for min χ^2	$f(\beta,x,y)$ for ML
Logistic $\theta = \frac{e^{\beta x}}{1+e^{\beta x}}$	$\log(\frac{\theta}{1-\theta})$	$\frac{e^{y\beta x}}{1+e^{\beta x}}$
Linear $\theta = \lfloor \beta x \rceil_0^1$	θ	$2(y-.5)\lfloor \beta x-.5 \rceil_{-.5}^{.5} + .5$
Cum. Normal $\theta = \Phi(\beta x)$	$\Phi^{-1}(\theta)$	$\Phi(2(y-.5)\beta x)$
Angular $\theta = \sin^2(\lfloor \beta x \rceil_0^{\pi/2})$	$\arcsin(\sqrt{\theta})$	$\sin^2(\lfloor \beta x \rceil_0^{\pi/2} + y\frac{\pi}{2})$

Table 1. Comparison of four models and two methods.

* $\lfloor f \rceil_a^b = \begin{cases} b & \text{if } f > b \\ f & \text{if } a \leq f \leq b \\ a & \text{if } f < a \end{cases}$

The rest of this paper considers the maximum likelihood method for the logistic model. BMDQLR, a program to do this, is now under development at Health Sciences Computing Facility, UCLA.

3. The input

The outcome variable y_j is recorded as 1 or 0, indicating success or failure respectively. The count n_j (the number of times the outcome is recorded for a given set of values of the explanatory variables) may also be recorded. Occasionally, instead of the outcome variable value, the number of successes s_j and the number of failures f_j for the n_j trials are recorded. To specify the outcome either y_j, n_j and s_j, n_j and f_j, or s_j and f_j must be recorded. An input case represents 1 or n_j outcomes.

The explanatory or covariate variables are

$v_1, v_2, \ldots, v_q$, the variables measured in interval scales, and

$g_1, g_2, \ldots, g_r$, the categorical variables.

The program generates a set of design variables for each categorical variable and each interaction term.

A categorical variable g_i with ℓ_i levels gives rise to a set of ℓ_i-1 design variable components:

$$d_i = (d_i^1, d_i^2, \ldots, d_i^{\ell_i - 1}).$$

These components could represent the linear, quadratic, etc., components, or they may only be contrasts between the ℓ_i groups. For the sake of computational speed we use components that contrast each of the first ℓ_i-1 groups with the last group. For example, for three categories the contrasts are (1,0,-1) and (0,1,-1).

A group of k categorical variables yeilds a kth order interaction, which in turn gives rise to a set of $\prod_{j=1}^{k} (\ell_{i_j} - 1)$ design components:

$$d_{i_1 i_2 \ldots i_k} = \prod_j d_{i_j} .$$

For example, for three categorical variables A, B and C with 2, 4 and 3 levels, seven sets of design variables are generated. The number of design variables in each of these sets is: 1 for A, 3 for B, 2 for C, 3 for AxB, 2 for AxC, 6 for BxC and 6 for AxBxC.

4. Building the model

The parameters $(\beta_i) = \beta$ of the model (1.2) are estimated to maximize the likelihood function (1.3), and are computed using the nonlinear iterative process proposed by Jennrich and Moore [7].

The computation proceeds in a stepwise manner. At each step an interval-scaled variable or a set of design variable components is entered or removed from the model. Design variables are only considered in sets such as d_i or $d_{i_1 i_2 \ldots i_k}$.

Interactions are considered in hierarchical order. An interaction is considered for entry into the model only if the model does not have a higher order interaction which contains it. For example, if A, B and C are categorical explanatory variables:

AxBxC enters only if A, B, C, AxB, AxC and BxC are all in the model;

A may be removed only if AxB, AxC and AxBxC are not in the model.

At each step the variables x in (1.2) and (1.3) consist of some interval-scaled variables v_i, some design variable component sets d_j, and some design variable sets representing interactions.

A step consists of entry of a term into the model, if its computed significance is greater than a user-specified limit, or removal of a term from the model if its computed significance is less than a user-specified limit [5].

The user can select one of two methods for estimating the significance of a term:

- The MLR method, using the significance of the χ^2 obtained from the log of the ratios of the maximized likelihood functions

$$\chi^2 = 2|\log(L(\hat{\beta}_{current})/L(\hat{\beta}_{candidate}))| \quad (4.1)$$

- The ACE method using the significance of the F value obtained from the asymptotic covariance matrix via the following steps:

  - Assume that the function

$$h(\beta x) = \frac{e^{\beta x}}{1+e^{\beta x}} \quad (4.2)$$

  is linear in a region containing $\hat{\beta}_{current}$ and $\hat{\beta}_{candidate}$.

  - Perform the necessary pivot (sweep) operations to step in or out the term to be tested.

  - Compute the "F-value" from the apparent change in the "residual sum of squares" (i.e. diagonal element corresponding to the outcome variable).

The ACE method is not exact, since the assumed linearity (4.2) does not hold. The utility of the ACE method is that it is considerably faster than MLR and that the rank order of significances computed by ACE is very highly correlated with that computed by MLR.

When the significances are approximated by the ACE method, the computed significance for entry of a term may not equal the significance computed after its entry. The potential problem of cycling a term in and out of the model is prevented by limiting the number of times a term may be moved.

5. Examples

The first example uses the data from the Morrison [9] study. The data are treated by Bishop et al. [2] and are also used as an example for multiway contingency tables (BMDP3F) by Dixon and Brown [4].

The three year survival of breast cancer patients (see Table 2) is the outcome variable. The data are grouped by diagnostic center, inflammation size and appearance, and are stratified into three age groups. Center, size and type are considered categorical, and age is treated as an interval-scaled variable. Unfortunately the exact age is not available, so we use the ordered strata codes as an interval-scaled variable.

			Minimal Inflammation		Greater Inflammation	
Diagnostic Center	Age	Survived	Malignant Appearance	Benign Appearance	Malignant Appearance	Benign Appearance
Tokyo	Under 50	No	9	7	4	3
		Yes	26	68	25	9
	50-69	No	9	9	11	2
		Yes	20	46	18	5
	70 or over	No	2	3	1	0
		Yes	1	6	5	1
Boston	Under 50	No	6	7	6	0
		Yes	11	24	4	0
	50-69	No	8	20	3	2
		Yes	18	58	10	3
	70 or over	No	9	18	3	0
		Yes	15	26	1	1
Glamorgan	Under 50	No	16	7	3	0
		Yes	16	20	8	1
	50-69	No	14	12	3	0
		Yes	27	39	10	4
	70 or over	No	3	7	3	0
		Yes	12	11	4	1

Table 2. Survival of breast cancer patients, Morrison et al..

For the program to consider a model with terms: CONSTANT, AGE, TUMOR, SIZE, CENTER, TUMOR x SIZE, TUMOR x CENTER, SIZE x CENTER, TUMOR x SIZE x CENTER we state in the MODEL statement the highest order interaction terms: MODEL=CONSTANT, AGE, TUMOR*SIZE*CENTER.

The BMDQLR input cards for this problem are:

```
/PROBLEM  TITLE IS 'SURVIVAL OF BREAST CANCER PATIENTS
                                  - MORRISON ET AL.'.
/INPUT    VARIABLES ARE 7.  FORMAT IS '(7F2.0)'.
/VARIABLE NAMES ARE NUMBER,TUMOR,SIZE,SURVIVAL,AGE,
                                  CENTER,CONSTANT.
/GROUP    CODES(4) ARE 1,2.  NAMES(4) ARE NO,YES.
/REGRESS  DEPENDENT IS SURVIVAL.  COUNT IS NUMBER.
          INTERVAL = AGE.
          MODEL IS CONSTANT, AGE, TUMOR*SIZE*CENTER.
/END
 9 1 1 1 1 1 1
 7 2 1 1 1 1 1
 4 1 2 1 1 1 1
 3 2 2 1 1 1 1
26 1 1 2 1 1 1
68 2 1 2 1 1 1
25 1 2 2 1 1 1
 9 2 1 2 1 1 1
 1 2 2 2 2 3 1
14 1 1 1 2 3 1
12 2 1 1 2 3 1
 3 1 2 1 2 3 1
27 1 1 2 2 3 1
39 2 1 2 2 3 1
10 1 2 2 2 3 1
 4 2 2 2 2 3 1
 3 1 1 1 3 3 1
 7 2 1 1 3 3 1
 3 1 2 1 3 3 1
12 1 1 2 3 3 1
11 2 1 2 3 3 1
 4 1 2 2 3 3 1
 1 2 2 2 3 3 1
```

Figure 1. Input cards for the breast cancer data.

The ACE method of term selection is used. If the MLR method had been desired, "METHOD=MLR." would have been included in the regression specification. The assumed limit for significance to enter a term is 0.1, to remove a term, 0.15.

At each step the following results are printed (circled numbers refer to those in the output in Figure 2).

(1) the log of the maximized likelihood value, the χ^2 value and its p-value, which measures the change of the log-likelihood value from the previous step, and the goodness-of-fit χ^2 value (and its p-value).

(2) for each component, the computed coefficients, their asymptotic standard errors and the ratio of coefficient to standard error

(3) the approximate F-to-enter or F-to-remove and their p-values for each term to be considered in the next step, and, if METHOD=MLR, the log-likelihood value for the model if the term were entered or removed

Note that for the example at step 3 (Figure 2) the SIZE x CENTER and TUMOR x SIZE are not considered because SIZE is not in the model.

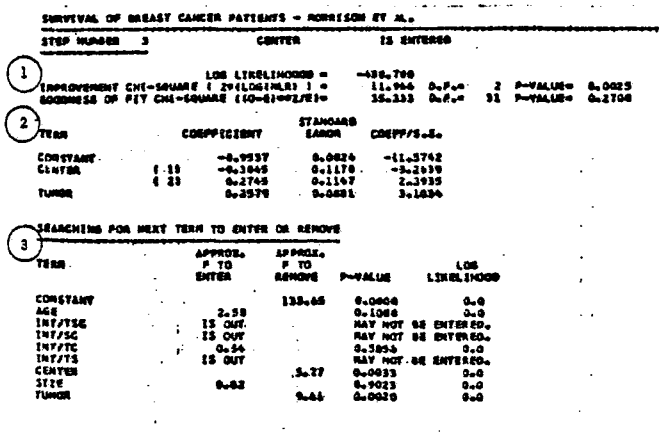


Figure 2. Sample output obtained from setup shown in Figure 1.

Table 3 illustrates the high correlation of significances computed by the ACE and MLR methods. The values were obtained from the breast cancer data run with both ACE and MLR methods.

P-ACE / P-MLR	Step 0 Enter	Step 0 Remove	Step 1 Enter	Step 1 Remove	Step 2 Enter	Step 2 Remove	Step 3 Enter	Step 3 Remove
Constant	.0000 .0000			.0000 .0000		.0000 .0000		.0000 .0000
Age	.0000 .0000		.0145 .0148		.0106 .0108		.1088 .1090	
Tumor	.0001 .0001		.0026 .0027			.0029 .0027		.0020 .0018
Size	.0000 .0000		.7360 .7364		.4670 .4649		.9023 .9021	
Center	.0002 .0002		.0040 .0036		.0028 .0025			.0033 .0025
Tumor x Size								
Tumor x Center							.5856 .5830	
Size x Center								
Tumor x Size x Center								

Table 3. Comparison of p-values for ACE and MLR.

Empty places in the table indicate that the corresponding term was not tested due to the hierarchical inclusion rule of interactions.

The simplicity of deck setups is illustrated with two more examples, both of which are from Cox [3].

```
/PROBLEM  TITLE='BRAND X - COX, PAGE 38-39'.
/INPUT    VARIABLES=6.   FORMAT='(6F4.0)'.
/VARIABLE NAMES ARE TOTAL,PREF,TEMP,USER,SOFTNESS,
                    CONSTANT.
/REGRESS  COUNT=TOTAL. SCOUNT=PREF. INTERVAL=SOFTNESS.
          MODEL=SOFTNESS,CONSTANT,TEMP*USER.
/END
 110  68   1.  0   3   1
  72  42   2   0   3   1
  89  37   1   1   3   1
  67  24   2   1   3   1
 116  66   1   0   2   1
  56  33   2   0   2   1
 102  47   1   1   2   1
  70  23   2   1   2   1
 116  63   1   0   1   1
  56  29   2   0   1   1
 106  57   1   1   1   1
  48  19   2   1   1   1
```

Figure 3. Input for Cox's Brand X preference data.

Figure 3 shows the deck setup for the preference of the new detergent X over the standard M. The three explanatory variables are water softness, an interval-scaled variable recorded as 1, 2 or 3, water temperature at two levels and a factor indicating previous use of brand M. Cox treats softness as a categorical variable. The deck setup in Figure 3 treats softness as an interval-scaled variable.

```
/PROBLEM  TITLE='INGOT DATA - COX, PAGES 68 AND 91'.
/INPUT    VARIABLES=5.   FORMAT='(5F5.0)'.
/VARIABLE NAMES ARE HEAT,SOAK,READY,NOTREADY,CONSTANT.
/REGRESS  SCOUNT=READY.  FCOUNT=NOTREADY.
          INTERVAL = HEAT,SOAK.
/END
 7.0  1.0  10.  0.   1.
 7.0  1.7  17.  0.   1.
 7.0  2.2   7.  0.   1.
 7.0  2.8  12.  0.   1.
 7.0  4.0   9.  0.   1.
14.0  1.0  31.  0.   1.
14.0  1.7  43.  0.   1.
14.0  2.2  31.  0.   1.
14.0  2.2   0.  2.   1.
14.0  2.8  31.  0.   1.
14.0  4.0  19.  0.   1.
27.0  1.0  55.  0.   1.
27.0  1.0   0.  1.   1.
27.0  1.7  40.  0.   1.
27.0  1.7   0.  4.0  1.
27.0  2.2  21.  0.   1.
27.0  2.8  21.  0.   1.
27.0  2.8   0.  1.   1.
27.0  4.0  15.  0.   1.
27.0  4.0   0.  1.   1.
51.0  1.0  10.      1.
51.0  1.0   0.  3.0  1.
51.0  1.7  1.0  0.   1.
51.0  2.2   1.  0.   1.
51.0  4.0   1.  0.   1.
```

Figure 4. Input for Cox's Ingot data.

The data in Figure 4 are the number of ingots ready to roll (first column) and the number not ready (second column) tabulated by the explanatory variables heating time and soaking time, both interval-scaled. The setup shown in Figure 4 yields the identical result shown in Cox [3, p. 91].

6. Summary

A stepwise logistic regression program, BMDQLR, will soon be available from Health Sciences Computing Facility, UCLA. This program can be used to build

logistic models. It requires a relatively simple set of instructions. The program

- handles categorical or interval-scaled explanatory variables,
- automatically generates interaction terms and design variable components for categorical explanatory variables,
- enforces a hierarchical inclusion rule of interaction terms
- provides a choice of term selection methods;
  ACE - fast but approximate
  MLR - slow but more exact

The output produced at each step, such as that illustrated in Figure 2, provides useful information concerning the explanatory variables.

At the conclusion of the stepping process the program prints the results in a number of different forms to enhance assessment of the results. These include:

- a summary table indicating which term was entered or removed at each step, the degrees of freedom of the term, the model's log-likelihood estimate at each step, the χ^2 measuring the change in the log-likelihood from step to step, and the goodness-of-fit χ^2.
- for each distinct pattern of the considered independent variables the number of cases in each of the two outcome groups, the proportion of the first group to the total, the predicted probability of the first group and the values of the independent variables
- scatter plots of the proportion of first group versus the predicted probability of first group and the proportion of first group versus the predicted log odds. These scatter plots are useful if patterns contain more than one case and proportions are not 0 or 1.
- for each distinct pattern of independent variables in the final model the above two are repeated.
- histograms of predicted probabilities for each group; these histograms are useful if there are interval-scaled independent variables in the model, resulting in computed probabilities that have more than a few discrete values.
- a table, summarizing the counts and percents of correct and incorrect predictions if probabilities at various cutpoints are used to classify cases into outcome groups
- a plot of percent correct classifications as a function of the cutpoint for each of the outcome groups and for the total sample

References

[1] Berkson, J. (1952). A statistically precise and relatively simple method of estimating the bioassay with quantal response, based on the logistic function. JASA 47, 565-599.

[2] Bishop, W.M.M., S.E. Feinberg and P.W. Holland (1975). Discrete Multivariate Analysis: Theory and Practice. Cambridge, Mass., MIT Press.

[3] Cox, D.R. (1970). Analysis of Binary Data. London, England. Methuen.

[4] Dixon, W.J. and M.B. Brown (1977). BMDP-77 Biomedical Computer Programs, P-Series. Berkeley, Calif., UC Press.

[5] Efroymsen, M.A. (1960). Multiple regression analysis. Mathematical Methods for Digital Computers. A. Ralston and H.S. Wilf, Eds. New York, Wiley.

[6] Finney, D.J. (1964). Statistical Methods in Biological Assay. London, Griffin. Chapter 17, pp. 437-467.

[7] Jennrich, R.I. and R.H. Moore (1975). Maximum likelihood estimation by means of nonlinear least squares. Statistical Computing Section Proceedings of the American Statistical Association.

[8] Lee, E.T. (1974). A computer program for linear logistic regression analysis. Computer Programs in Biomedicine 4, 80-92.

[9] Morrison, A.S., M.M. Black, C.R. Lowe, B. MacMahon and S.Y. Yuasa (1973). Some international differences in histology and survival in breast cancer. Int'l. J. Cancer 11, 261-267.

[10] O'Neill, T.J. (1975). A comparison of logistic regression and maximum likelihood classification methods. Technical Report No. 12, Stanford University, Division of Biostatistics, Stanford.

[11] Prentice, R. (1976). Use of logistic model in retrospective studies. Biometrics 32, 599-606.

METHODS IN BMDP FOR DEALING WITH ILL-CONDITIONED DATA -- MULTICOLLINEARITY AND MULTIVARIATE OUTLIERS

James W. Frane
Health Sciences Computing Facility, UCLA

Abstract

A statistical analysis is ill-conditioned if it is overly sensitive to the input (i.e., a small change in the input may cause a large change in the results of the analysis) or if it is difficult to compute a numerically correct answer. One kind of ill-conditioning is due to variables (multicollinearity); another to individual cases or observations (multivariate outliers). Checks are built into BMDP-77 programs to prevent an investigator from trying to answer questions that are very ill-conditioned in the numerical sense; the regression programs also describe the nature of any numerical ill-conditioning. The new BMDP9R regression program provides several statistics that describe the sensitivity of the analysis to the individual cases.

Introduction

We define a statistical analysis as ill-conditioned if it is overly sensitive to the input in the sense that a small change in the input may cause a large change in the results of the analysis. There are two types of ill-conditioning: one is due to variables (multicollinearity), and the other to individual cases (multivariate outliers). A problem of the first type is numerically ill-conditioned when it is difficult to obtain a numerically accurate solution. Problems of the second type are statistically ill-conditioned. Numerically ill-conditioned problems are frequently statistically ill-conditioned also.

We are concerned with total accuracy -- statistical and numerical. Therefore, both types of ill-conditioning must be considered. Unless the fit to a regression equation is extremely good, statistical accuracy is likely to be more difficult to obtain than numerical accuracy and statistical error is likely to be larger than numerical error. As demonstrated by Frane (1974), by using a single pass of the data and accumulating cross products of deviations in single precision, it is possible to obtain estimates of the regression coefficients for the highly ill-conditioned Longley (1967) data such that the ratio of numerical computing error in each coefficient divided by the corresponding standard error is .0005 or less.

For either type of ill-conditioning, it is quite likely that the original ill-conditioned analysis should be replaced or supplemented by a well-conditioned analysis. BMDP-77 statistical computer programs provide a number of methods for describing the nature of the ill-conditioning, and a variety of methods for obtaining a well-conditioned analysis.

This paper describes the available methods. The BMDP-77 manual edited by Dixon and Brown (1977) describes how to use the programs. BMDP programs are updated constantly, so versions dated before December 1977 do not contain all the features described here.

1. Detecting and Describing Numerical Ill-conditioning

BMDP-77 includes several programs for regression that focus on specialized needs. Some use single precision computation, some double precision. All programs read data and perform transformations (if any) in single precision. All programs have some kind of tolerance test to detect multicollinearity.

A basic method in most of the BMDP-77 linear and nonlinear regression programs is the accumulation of cross products of deviations by a provisional means algorithm. Typically, this matrix is swept (or pivoted) under control of a tolerance test. The tolerance test used until August 1976 was the classical Efroymsen (1960) test. Berk (1976), at the Ninth Interface Symposium, showed that the Efroymsen test was, at least theoretically, unsatisfactory in the context of classicial stepwise regression. In BMDP, variables in linear regression are always selected in a stepwise fashion: no variable is included if its squared multiple correlation with already included variables exceeds one minus the tolerance limit, or if including that variable causes the squared multiple correlation of an already included variable with the other included variables to exceed one minus the tolerance limit (Frane, 1977). In BMDP linear regression programs, whenever a variable is not included because it fails the tolerance test a message is written that it failed the test, and the tolerance for that variable is given.

The provisional means method protects against small coefficients of variation, but does not protect against extremely small coefficients of variation, except in the programs that use double precision for computations.

Data are always represented in single precision in BMDP. It has been frequently stated (or at least suggested) that single precision numbers have seven decimal digits of numerical accuracy on IBM 360 and 370 computers. A more accurate statement would be six digits: a number with a nonbinary fractional part has 24 bits in its mantissa but it may have as few as 21 bits of accuracy since the leading three bits could be zero. Twenty-one bits give slightly more than six decimal digits. Integer-valued real numbers can be represented without numerical error if they do not exceed 2^{24}. If a problem actually requires data entry beyond these levels, modifications can be made to the BMDP source.

Sometimes data appear to have more digits of accuracy than they have in fact. This can be deceptive in ill-conditioned analyses. Economic time series (such as that used by Longley, 1967) frequently yield numerically ill-conditioned analyses. A frequent misconception is that in these series time is measured without error. For example, in the Longley data time is measured by year. However, the year is not a perfect measure of time because of leap years. Table 1 contains the regression coefficients for the Longley data in original form and after year has been converted to "true" time by adjusting for leap years. Frane (1974) and Beaton, Rubin and Barone (1976) give a variety of other statistical reasons for avoiding numerically ill-conditioned analyses.

Each BMDP linear regression program provides some information about ill-conditioning, but if ill-conditioning is known to be a problem it is advisable to use BMDP9R because computations in this program are performed in double precision. Figure 1 contains selected portions of the output when BMDP9R is used to analyze the Longley data (the first variable has been multiplied by 10 to avoid fractional values). There are several indications of possible ill-conditioning in the analysis. Note the small coefficient of variation (standard deviation divided by mean) for the year. The tolerance for each independent variable is one minus its squared multiple correlation with the other independent variables. The reciprocal of the tolerance for an independent variable is called the variance inflation factor because the variance of its regression coefficient is inversely proportional to its tolerance. Thus, numerically ill-conditioned analyses frequently have problems with statistical accuracy so the tolerance check is both a numerical and a statistical safeguard. Note that the tolerance for GNP is .0006; the multiple correlation of GNP with the other independent variables is .9997. All digits in the reported regression equation are numerically accurate. The next six digits (not printed) are also numerically accurate. However, since there is little statistical accuracy in the coefficients, the number of useful digits is far less than the number of numerically accurate digits.

When you recognize that an analysis is numerically ill-conditioned, there are several alternative analyses you may wish to perform. You can do some form of stepwise regression (P2R), or you can consider an analysis using one or a wide variety of possible subsets of the data (P9R).

A different approach is to do a regression on some, but not all, principal components (P4R). P4R reports the eigenvalues of the correlation matrix of the independent variables. An eigenvalue lower limit is included. This limit corresponds to a tolerance test, and the limit should not ordinarily be set lower than its default value. Eigenvalues of the correlation matrix are usually used rather than those of the covariance matrix because the units of measurement of the independent variables are usually arbitrary and the numerical methods used in BMDP are not sensitive to the scale of the independent variables. Similarly, the correlation matrix is used rather than some sort of matrix of cross products computed without subtracting means because (1) the zero point on the measurement scale is frequently arbitrary (e.g., year measured A.D. or from some other point in time), (2) the numerical methods are not very sensitive to moderately small coefficients of variation, and (3) a small coefficient of variation is usually an artificial form of ill-conditioning, which is easily eliminated by subtracting values roughly equal to the mean; e.g., subtracting 1900 or 1950 from the year in the Longley data.

In the Longley data, two eigenvalues are less than the default eigenvalue limit, so only four principal components are used. For greater detail P4R can be supplemented by a detailed principal components analysis of the independent variables, usually with rotation (P4M). Rotation is suggested because it is often easier to semantically interpret the rotated loadings than the unrotated loadings. P4M includes (among other things) eigenvalues, rotated and unrotated factor loadings, factor (component) scores, plots of rotated and unrotated loadings, and factor score plots. When the number of variables is large, the shaded correlation matrix is especially helpful. (Although the P4M output for the Longley data is interesting, space does not permit us to include it here.)

In an economic time series many variables are highly correlated with time. It may be interesting to compute the partial correlation matrix for the dependent and independent variables after removing the linear effects of time (P6R). This partial correlation matrix is computed by the process of regressing the dependent and independent variables on time. Both the partial covariance matrix (equivalent to partial correlation matrix) and the raw data plus residuals can be output to a BMDP file. The BMDP file can then be used as input to other regression routines. Alternatively, the stepwise regression program (P2R) can be used to force time into the equation first.

P9R also reports a numerical consistency check: the residual mean square is computed both as part of the computation of the regression equation and from the residuals themselves.

Canonical correlation (P6M) can be viewed as an extension of regression for two sets of variables. For each of the two sets of variables BMDP reports the squared multiple correlation of each variable with all other variables in its set. The tolerance test prevents any variable from being included in the analysis if its squared multiple correlation with the other variables in its set exceeds one minus the tolerance limit. Figure 2 shows part of the results from P6M for the Longley data when the first set of variables is the usual set of independent variables and the second set contains some of the components of derived employment (the usual dependent variable). P6M also reports a numerical consistency check: the sample variance of the canonical variable scores should be equal to one; the actual variance is computed and reported, together with the relative error.

Figure 1

Univariate Statistics and Regression Coefficients from BMDP9R

UNIVARIATE SUMMARY STATISTICS

VARIABLE	MEAN	STANDARD DEVIATION	COEFFICIENT OF VARIATION
1 GNPIPDEF	1016.81250	107.91553	0.106131
2 GNP	387698.43750	99394.93780	0.256372
3 UNEMPLOY	3193.31250	934.46425	0.292632
4 ARMFORCE	2606.68750	695.91960	0.266975
5 NONINPOP	117424.00000	6956.10156	0.059239
6 YEAR	1954.50000	4.76095	0.002436
7 DEREMPLO	65317.00000	3511.96836	0.053768

SMALLEST VALUE	LARGEST VALUE	SMALLEST STANDARD SCORE	LARGEST STANDARD SCORE	SKEWNESS	KURTOSIS
830.00000	1169.00000	-1.73	1.41	-0.13	-1.40
234289.00000	554894.00000	-1.54	1.68	0.02	-1.35
1870.00000	4806.00000	-1.42	1.73	0.14	-1.30
1456.00000	3594.00000	-1.65	1.42	-0.37	-1.20
107608.00000	130081.00000	-1.41	1.82	0.26	-1.27
1947.00000	1962.00000	-1.58	1.58	0.0	-1.43
60171.00000	70551.00000	-1.47	1.49	-0.09	-1.55

SQUARED MULTIPLE CORRELATION	0.99548
MULTIPLE CORRELATION	0.99774
ADJUSTED SQUARED MULT. CORR.	0.99247
RESIDUAL MEAN SQUARE	0.929360D 05
STANDARD ERROR OF EST.	0.304854D 03
F-STATISTIC	330.29
NUMERATOR DEGREES OF FREEDOM	6
DENOMINATOR DEGREES OF FREEDOM	9
SIGNIFICANCE	0.0000

VARIABLE NO.	NAME	REGRESSION COEFFICIENT	STANDARD ERROR	STAND. COEF.	T-STAT.	2TAIL SIG.	TOLERANCE
	INTERCEPT	-0.348226D 07	0.8904200 06	-991.540	-3.91	0.004	
1	GNPIPDEF	0.150619D 01	0.849149D 01	0.046	0.18	0.863	0.007378
2	GNP	-0.358192D-01	0.334910D-01	-1.014	-1.07	0.313	0.000559
3	UNEMPLOY	-0.202023D 01	0.488400D 00	-0.538	-4.14	0.003	0.029745
4	ARMFORCE	-0.103323D 01	0.214274D 00	-0.205	-4.82	0.001	0.278635
5	NONINPOP	-0.511041D-01	0.226073D 00	-0.101	-0.23	0.826	0.002505
6	YEAR	0.182915D 04	0.455478D 03	2.480	4.02	0.003	0.001318

Figure 2

Selected Output from BMDP6M

THERE ARE 6 VARIABLES IN THE SECOND SET OF VARIABLES
BUT THE EFFECTIVE RANK IS ONLY 4

NON-PIVOTED VARIABLES ARE

NUMBER	NAME	SQUARED MULTIPLE CORRELATION WITH PIVOTED VARIABLES
2	GNP	0.99844
6	YEAR	0.99631

SQUARED MULTIPLE CORRELATIONS OF EACH VARIABLE IN SECOND SET WITH ALL OTHER VARIABLES IN SECOND SET (EXCLUDING NON-PIVOTED VARIABLES)

VARIABLE NUMBER	NAME	R-SQUARED
1	GNPIPDEF	0.97220
3	UNEMPLOY	0.68230
4	ARMFORCE	0.59965
5	NONINPOP	0.97109

SQUARED MULTIPLE CORRELATIONS OF EACH VARIABLE IN FIRST SET WITH ALL OTHER VARIABLES IN FIRST SET

VARIABLE NUMBER	NAME	R-SQUARED
8	AGREMPLO	0.91975
9	SELFEMPL	0.68579
10	UNPAMWK	0.91463
11	DOMESTIC	0.93781
12	NONAGRJB	0.94118

2. Detecting and Describing Statistical Ill-Conditioning

Frequently an analysis is overly sensitive to the data for a single case (row of the data matrix) because the case is an outlier on a single variable. As an integral part of the regression analysis, P2R and P9R report the largest and smallest values for each variable and the largest and smallest standardized scores. When some values are missing, it is important that the extreme values be reported as part of the regression analysis. If they are reported only as part of a univariate screening procedure, all values available for each variable are usually used, rather than values for only the cases that are complete (have values for all variables used in the analysis). P2R and P9R eliminate any case that has missing values for any of the variables used. Thus the extreme values reported by these programs may differ from those reported in routines that do only univariate screening.

Identification of univariate outliers is a relatively easy and straight-forward task. A more difficult problem is to identify and describe cases that are multivariate outliers. An index of the influence each case has on the regression equation is given by Cook (1977). Cook's index is included in the April 1977 update to BMDP9R, and can be defined and interpreted in a number of ways. Our preferred way is as follows: For each case, Cook's index is a monotonically increasing function of both the Mahalanobis distance from the case to the center of all cases in the space of the independent variables, and of the standardized (Studentized) residual for that case. Thus a case has a large influence if

a) its Mahalanobis distance is large,

b) its standardized residual is large, or

c) both the Mahalanobis distance and standardized residual are somewhat large.

For each case, P9R reports Cook's index, the Mahalanobis distance, the standardized residual, and the standard error of the predicted value. Figure 3 shows output from P9R for the Longley data. Note that the data for the year 1951 are very influential (Frane, 1974, identified this fact by means of Tukey's catchers).

Once a case has been identified as highly influential, several strategies can be used. P9R automatically updates the regression coefficients by removing the most influential case, and reports the original and updated coefficients side-by-side. For the Longley data (Table 1), the most drastic change is for the intercept, which changed by four standard errors and added an opposite sign. This can easily happen when the regression coefficient is nonsignificant, but in the original and transformed data, the t statistic for the intercept is highly significant.

Interesting insights to help interpret data can be gained by computing the residual for each case after removing the effect of the case from the regression equation; these deleted residuals can be compared in a bivariate plot in P9R. The deleted residuals are also used to define David Allen's (1971) prediction sum of squares (PRESS). In the Longley data, the deleted residuals are large for both 1951 and 1962 -- much larger than the ordinary residuals; this is an indication of the sensitivity of the analysis to the data. The error mean square from a regression is an estimate of the population residual variance, σ^2. The sum of squared deleted residuals (prediction sum of squares) divided by the number of cases is also an estimate of σ^2. For the Longley data, the ratio of these estimates of σ^2 is about two. Unfortunately, we do not know the statistical properties of this ratio except that it should be approximately one.

Suppose we have a case with a large value for Cook's index. If it has a large standardized residual but its Mahalanobis distance is not large, it is likely that the influence is due to an unusual (perhaps incorrectly recorded) value for the dependent variable.

If the standardized residual is small, the influence is due to its Mahalanobis distance. In this case it is possible that one or more values of the independent variables are unusual or incorrectly recorded. To identify these values we can perform a stepwise discriminant analysis (P7M) in which the outlier case in question is group one and all cases that are not considered outliers or unduly influential are in group two. Since a two-group discriminant analysis can be performed as a regression analysis with a binary dependent variable, P9R (which considers all possible subsets) can also be used for this analysis. Weisberg (1977) has noted that the data set given in Narula and Wellington (1977) has one case with a very large effect on the regression equation -- it is a bivariate (not a univariate) outlier in the vector space of the independent variables. Both Cook's index and the Mahalanobis distance are large.

If the standardized residual for an influential case is moderate or large and its Mahalanobis distance is moderate or large, the above discriminant analysis can be performed using both independent and dependent variables. This is the case for the years 1951 and 1962 in the Longley data and for the oxidation of ammonia data in Daniel and Wood (1971, p. 61).

3. Nonlinear Regression

Nonlinear regression problems have all the numerical and statistical difficulties found in linear regression, but are particularly difficult when constraints are imposed on the parameters to be estimated.

There are two BMDP programs for nonlinear regression. BMDP3R uses a modified Gauss-Newton approach; this requires user-specified derivatives. BMDPAR uses a secant approach and does not use derivatives. PAR was described by Ralston at last year's symposium and is discussed in Ralston and Jennrich (1978).

Similar to the BMDP linear regression programs, the nonlinear programs sweep matrices of cross products under control of a tolerance test. BMDP3R was originally written in single precision, but both programs now do computations in double precision. P3R was recently updated by Jennrich and Sampson to improve its performance on problems with constraints on the parameters.

P3R and PAR both handle boundary constraints on parameters such as

$$a_i \le p_i \le b_i$$

In P3R, you can impose additional constraints of the form $\Sigma c_i p_i = d$. For example, in P3R you can solve problems with constraints $0 \le p_i \le 1$ $(i=1,\ldots,4)$ and $p_1 + p_2 + p_3 + p_4 = 1$, which cannot be handled in routines that permit only boundary constraints.

PAR handles general linear inequality constraints of the form

$$a \le \Sigma c_i p_i \le b, \quad (a \ne b)$$

CASE LABEL	CASE NO.	OBSERVED DEKEMPLO	PREDICTED VALUE	STANDARD ERROR OF PRED.VAL.	RESIDUAL	CASE WEIGHT	WEIGHTED RESIDUAL	STAND-ARDIZED RESIDUAL	DELETED (PRESS) RESIDUAL	MAHALA-NOBIS DISTANCE	COOK'S DISTANCE
1947	1	60323.0000	60055.6562	198.6322	267.3398	1.000	267.3398	1.16	464.5651	5.43	0.14
1948	2	61122.0000	61216.0117	229.1437	-94.0139	1.000	-94.0139	-0.47	-216.1132	7.54	0.04
1949	3	60171.0000	60124.7109	183.4389	46.2872	1.000	46.2872	0.19	72.5589	4.49	0.00
1950	4	61147.0000	61597.1133	185.9929	-410.1145	1.000	-410.1145	-1.70	-653.2857	4.65	0.24
1951	5	63221.0000	62911.2852	239.1718	309.7144	1.000	309.7144	1.64	805.5228	8.30	0.61
1952	6	63639.0000	63888.3086	185.3286	-249.3112	1.000	-249.3112	-1.03	-395.4644	4.61	0.09
1953	7	64989.0000	65153.0469	213.7311	-164.0490	1.000	-164.0490	-0.75	-322.6335	6.44	0.08
1954	8	63761.0000	63774.1797	216.5658	-13.1804	1.000	-13.1804	-0.06	-26.6085	6.63	0.00
1955	9	66019.0000	66004.6875	206.1131	14.3048	1.000	14.3048	0.06	26.3496	5.92	0.00
1956	10	67857.0000	67401.5625	175.2885	455.3940	1.000	455.3940	1.83	680.3174	4.02	0.24
1957	11	68169.0000	68186.2500	182.9824	-17.2689	1.000	-17.2689	-0.07	-26.9777	4.46	0.00
1958	12	66513.0000	66552.0000	211.8953	-39.0550	1.000	-39.0550	-0.18	-75.5598	6.31	0.00
1959	13	68655.0000	68810.5000	186.5120	-155.5500	1.000	-155.5500	-0.65	-248.6049	4.68	0.04
1960	14	69564.0000	69649.6250	145.6866	-85.6713	1.000	-85.6713	-0.32	-111.0276	2.49	0.00
1961	15	69331.0000	68989.0625	186.1534	341.9314	1.000	341.9314	1.42	545.2326	4.66	0.17
1962	16	70551.0000	70757.7500	252.9765	-206.7578	1.000	-206.7578	-1.22	-663.9933	9.39	0.47

SUMMARY STATISTICS FOR RESIDUALS

(CASES WITH POSITIVE WEIGHT)

AVERAGE RESIDUAL	0.0000
RESIDUAL MEAN SQUARE	92936.00616734
AVERAGE DELETED RESIDUAL	-9.1076
AVE. SQUARED DELETED RESIDUAL	180430.78384063

9.39 IS THE MAXIMUM VALUE OF MAHALANOBIS DISTANCE AMONG CASES WITH POSITIVE CASE WEIGHT. THIS OCCURRED FOR CASE NUMBER 16, CASE LABEL = 1962

1.83 IS THE LARGEST STANDARDIZED RESIDUAL (IN ABSOLUTE VALUE) AMONG CASES WITH POSITIVE CASE WEIGHT. THIS OCCURRED FOR CASE NUMBER 10, CASE LABEL = 1956

0.61 IS THE MAXIMUM VALUE OF COOK'S DISTANCE AMONG CASES WITH POSITIVE WEIGHT. THIS OCCURRED FOR CASE NUMBER 5, CASE LABEL = 1951
IF THIS CASE WERE OMITTED, THE REGRESSION COEFFICIENTS WOULD MOVE FROM THE VALUES REPORTED ABOVE TO THE EDGE OF A 28.45 PERCENT CONFIDENCE ELLIPSOID.

Figure 3
Residual Analysis from BMDP9R

Table 1

Comparison of Regression Coefficients

	Original	After Correcting Time	Omitting Data for Year 1951
intercept	3482260	77751.7	-496270
GNPIPD	1.50619	1.48250	3.16114
GNP	-.0358192	-.0352025	-.0837701
Unemployment	-2.02023	-2.01598	-2.69785
Size of Armed Forces	-1.03323	-1.03604	-1.25585
Noninstitutional Population	-.0511041	-.0567546	.166137
Year (Time)	1829.15	1824.16	2583.58

For example, PAR can solve problems with the constraints $0 \leq p_i \leq 1$ (i=1,2), and $0 \leq p_1 + p_2 \leq 1$; routines that permit only boundary constraints cannot handle these analyses.

P3R and PAR report residuals and the standard errors of the predicted values. Data for points with large standard errors for predicted values should be reviewed carefully. Other indicators of the sensitivity of the results to the individual data points could be computed, such as deleted residuals and Cook's index, available in P9R.

Other features in the BMDP-77 versions of P3R and PAR that are not always found in nonlinear least squares routines include case weights, user specification of loss function, and user specification or error mean square to be used for computing asymptotic standard errors. These features are frequently used in conjunction with maximum likelihood estimation and testing. Case weights can be redefined at each iteration in order to do iteratively reweighted least squares. Two passes of the data can be requested for each iteration (e.g., when fitting multinomial models as explained in the 1977 manual).

4. Conclusion

The accuracy of computer programs is a sensitive issue. In BMDP, we are concerned with both numerical and statistical accuracy. An analysis is incomplete if it does not describe the nature of any numerical or statistical ill-conditioning.

References

1. Allen, D.M. (1971). The prediction sum of squares as a criterion for selecting predictor variables. Univ. Of Kentucky, Dept of Statistics, Technical Report No. 23.

2. Beaton, A.E., D.B. Rubin and J.L. Barone (1976). The acceptability of regression solutions: Another look at computational accuracy. Jour. Amer. Statist. Assoc. 71, 158-168.

3. Berk, K.N. (1976). Tolerance and conditions in regression computations. In Proceedings of the Ninth Interface Symposium on Computer Science and Statistics, Boston, Prindle, Weber and Schmidt, 202-203.

4. Cook, R.D. (1977). Detection of influential observations in linear regression. Technometrics 19, 15-18.

5. Daniel, C. and F.S. Wood (1971). Fitting Equations to Data, New York, Wiley.

6. Dixon, W.J. and M.B. Brown, Eds. (1977). BMDP-77 Biomedical Computer Programs, P-Series, Berkeley, Univ. of Calif. Press.

7. Efroymsen, M.A. (1960). Multiple Regression Analysis. Mathematical Methods for Digital Computers I, New York, Wiley, 191-203.

8. Frane, J.W. (1974). The role of numerically ill-conditioned data in the evaluation of computer programs for least squares. (Presented at the annual meeting of the American Statistical Association.) Health Sciences Computing Facility Technical Report No. 7.

9. Frane, J.W. (1977). A note on checking tolerance in matrix inversion and regression. Technometrics 19, 513-514.

10. Longley, J.W. (1967). An appraisal of least squares programs for the electronic computer from the point fo view of the user. Jour. Amer. Statist. Assoc. 62, 819-841.

11. Narula, S.C. and J.F. Wellington (1977). Prediction, linear regression and the minimum sum of relative errors. Technometrics 19, 185-190.

12. Ralston, M.L. and R.I. Jennrich (1978). Dud, a derivative-free algorithm for nonlinear least squares. Technometrics, February (in press).

13. Weisberg, S. (1977). A statistic for allocating C_p to individual cases. School of Statistics, Univ. of Minnesota, Technical Report No. 296.

BEST SUBSETS REGRESSION UNDER THE MINIMAX CRITERION

James E. Gentle and W. J. Kennedy
Iowa State University

ABSTRACT

The use of a minimax (L_∞ or Chebyshev) criterion for estimation in a multiple linear regression model is considered. A procedure is described for selecting from among m potential regressors the best sets of each size for fitting the model under the minimax criterion. The method is a search that allows implicit enumeration by bounding.

1. INTRODUCTION

In recent years a number of criteria other than least squares have been suggested for estimation in the regression model

$$\underline{y} = X\underline{\beta} + \underline{\varepsilon}, \qquad (1)$$

where $\underline{y}$ is an n-vector of observations
X is an $n \times m$ matrix of observations
$\underline{\beta}$ is an m-vector of parameters to be estimated
and $\underline{\varepsilon}$ is an n-vector of random disturbances.

The choice of a resonable criterion for fitting depends on the distribution of $\underline{\varepsilon}$. If the distribution of $\underline{\varepsilon}$ is well-behaved and/or if a linear estimate of $\underline{\beta}$ is required, then least squares is an optimal criterion. Departures from these ideals, however, may suggest alternative criteria for fitting the linear model. Many situations encountered in practical data analysis suggest the use of methods that protect against heavy-tailed error distributions and/or arbitrarily deviant outliers in the data. In other cases the data do not appear to have an infinite-range distribution about the fitted model. Rice and White (1964) and Harter (1972) have suggested three minimum L_p estimators for use in the three situations alluded to above. In the case of identically, independently, normally distributed errors the L_2 or least squares estimator is the logical choice. For errors from heavier-tailed distributions, the L_1 or least absolute values estimator is recommended. When the errors are restricted in range, and particularly if the distribution is fairly uniform over the finite range, the L_∞ or minimax estimator has desirable properties. Harter (1972) suggested an adaptive procedure for selecting among these three estimators for use with a given data set.

2. ALGORITHMS FOR MINIMAX ESTIMATION

The problem of obtaining the minimax or L_∞ estimators for the model (1) may be posed as a linear program, as was pointed out by Kelly (1958). The primal linear programming problem is

$$\begin{aligned} &\text{minimize} && d \\ &\text{subject to} && X\underline{b} + d\underline{1}_n \geq \underline{y} \\ & && -X\underline{b} + d\underline{1}_n \geq -\underline{y}, \end{aligned} \qquad (2)$$

where $\underline{1}_n$ is an n-vector of 1's, and $\underline{b}$ is unrestricted. (d is ≥ 0 by the nature of the problem.) Kelley (1958) suggested using the dual formulation of the problem, and a number of modified dual simplex algorithms have been suggested for efficiently obtaining the solution. The straightforward dual simplex algorithm is very

inefficient both in storage and computation time relative to some special purpose methods available.

Barrodale and Phillips (1974) give one of the most efficient algorithms for obtaining the minimax estimates. Using the dual formulation,

$$\begin{aligned} \text{maximize} \quad & z = \underline{y}'(\underline{s} - \underline{t}) \\ \text{subject to} \quad & X'(\underline{s} - \underline{t}) = \underline{0} \\ & 1_m'(\underline{s} + \underline{t}) \leq 1 \\ & \underline{s}, \underline{t} \geq 0, \end{aligned} \tag{3}$$

they add a slack variable in the inequality constraint and an artificial variable in each row of the equality constraints. The final marginal costs of these variables are the optimal $\underline{b}$ in (2), that is, the minimax estimators. The relationships among variables are such that only n columns are required.

The algorithm begins by bringing the vectors associated with the s_i into the basis, replacing those corresponding to the artificial variables. The pivot column is chosen to correspond to the s_i with marginal cost of greatest absolute value, and the row is chosen so the pivot has largest absolute value among those corresponding to artificial variables in the basis.

Next the slack variable is driven out, using the pivot column corresponding to the s_i or t_i with marginal cost of greatest absolute value. (s_i is used if the marginal cost is negative; t_i, if negative.) It may be necessary to perform row interchanges and interchanges of s_i's and t_i's to be able to take the slack variable out.

The final stage of the algorithm uses as pivot column that one corresponding to the variable s_i or t_i having the most negative marginal cost. The pivot row is chosen by the usual selection rule of the simplex method. The procedure is continued until all marginal costs are nonnegative.

Barrodale and Roberts (1975) give a FORTRAN program for this algorithm.

3. PROCEDURE FOR BEST SUBSETS UNDER THE MINIMAX CRITERION

In the process of fitting equations to data, it is natural to seek the best subsets of explanatory variables for forming submodels which may adequately describe the relationships among the variables. This problem has been studied extensively under the least squares criterion. Various branch-and-bound routines such as the one by LaMotte and Hocking (1970) have been given to search efficiently among the subsets of each specified size to identify that subset giving the best least squares fit. Roodman (1974) described a branch-and-bound procedure for obtaining the best subsets of each size under the L_1 or least absolute values criterion. The key idea in each of these procedures is the effective use of fathoming. As many subsets as possible are eliminated from consideration without explicitly fitting those submodels. We now consider a branch-and-bound or partial enumerative scheme for selecting the best subsets under the L_∞ or minimax criterion. Although the implementation is different, the basic ideas are similar to those of Roodman(1974), which derive from procedures suggested by Balas (1965) and Geoffrion (1967). Some techniques of the implementation are similar to those described by Narula and Wellington (1977) for L_1 estimation. The basic program for obtaining the minimax estimates is that of Barrodale and Phillips (1975).

At a given stage of the algorithm each potential regressor will be in one of three possible states: fixed in, that is, of necessity included in the model; fixed out, that is, not considered for inclusion in the model; and free. The basic approach is to include all free variables, do a minimax fit, update states of potential regressors, and repeat.

The procedure follows a simple search tree discipline in which the root represents all variables in the model and from each node there are as many branches emanating as will yield unique subsets, assuming a search from the right. This is a commonly employed tree discipline, as, for example, in Narula and Wellington (1977). Figure 1 illustrates the tree and labeling scheme.

The labeling procedure as in Narula and Wellington (1977) will be employed here. The labels consist of the integers 1,2,...,m, a dot, and underlines. The root of the tree is labeled as a dot followed by all the integers, i.e., $\cdot 123\cdots m$. A forward step at a node is taken by moving the dot one space to the right. When all integers are to the left of the dot, the node represented is a terminal node. A backward step at a node is taken by underlining the rightmost integer not yet underlined to the left of the dot, moving the dot to the immediate right of this integer, and removing the underlines of any integers to the right of the dot in its new position. The labels on the nodes in Figure 1 represent a simple traversal of the tree. At any stage all integers to the left of the dot which are not underlined represent variables which are fixed out of the model.

We use the notation Z_j to represent a current

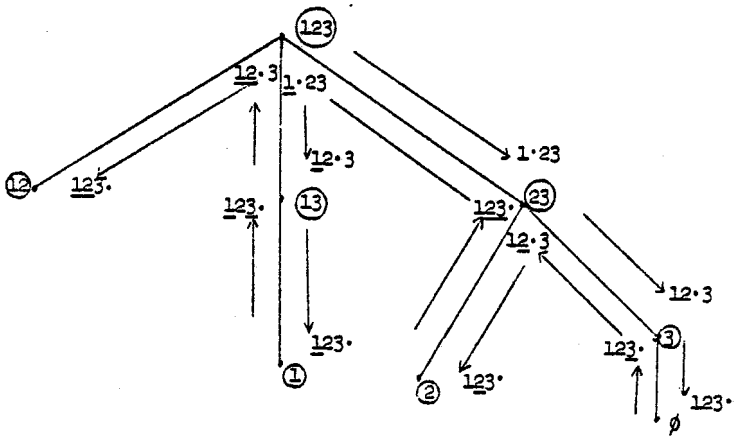


Figure 1. The Tree-Labeling Scheme

optimal solution and Z_j^* to represent the smallest currently known optimal value of Z in (3) for previous models containing j variables (regressors). (Note that the number of variables in the linear program remains constant, but the number of constraints varies as the number of regressors does.) We begin by obtaining Z_0^* (as the max $|y_i|$ or, if each submodel is to contain an intercept, the midrange of the y_i's) and Z_m^* (representing the full model). Initially all Z_j^* for $1 \leq j \leq m-1$ are assigned the value Z_0^*. Beginning at the root (with Z_m^* and the label ·123...m), update the label consistent with the search procedure (for example, the right-to-left procedure as shown in Figure 1) and go to Step 1.

Step 1: Let j be the number of variables in the model and let v be the number of integers underlined in the label or 1 if none are underlined. If j = 0 go to Step 5; otherwise go to Step 2.

Step 2: Obtain the solution for the current problem. (If Step 6 has been executed more recently than Step 5, the full solution may be avoided by first checking the increase in the objective function due to deletion of the regressor, which is equivalent to removal of a constraint. If the increase indicates Z_j will be greater than Z_v^* go to Step 5 or if Z_j will be greater than Z_j^* go to Step 4.) If $Z_j > Z_v^*$ go to Step 5; otherwise go to Step 3.

Step 3: If $Z_j < Z_j^*$ update the optimal solution of size j. Go to Step 4.

Step 4: If there are any integers in the label to the right of the dot go to Step 6; otherwise go to Step 5.

Step 5: If all integers in the label are underlined, terminate; otherwise backtrack to a new subset and go to Step 1.

Step 6: Take a forward step in the tree (i.e., update the label by moving the dot one space to the right) and go to Step 1.

This procedure can be modified so as to force a constant (intercept) to remain in the model at all times. Another simple modification is for finding the best subsets of sizes p, p + 1, p + 2, ..., q, where $\leq p \leq q \leq m$.

The fathoming in Step 2 does not appear to be successful as often as a similar fathoming in L_1 or least squares regressions.

REFERENCES

Balas, E. (1965). An additive algorithm for solving linear programs with zero-one variables. Operations Research 13, 517-546.

Barrodale, I. and Phillips, C. (1974). An improved algorithm for discrete Chebyshev linear approximations. Proceedings of Fourth Manitoba Conference on Numerical Mathematics. (H. C. Williams and B. L. Hartnell, Eds.) Winnipeg: Utilitas Mathematica Publishing Inc. 177-190.

Barrodale, I. and Phillips, C. (1975). Solution of an overdetermined system of linear equations in the Chebyshev norm. Transactions on Mathematical Software 1, 264-270.

Draper, N. R. and Smith, H. (1966). Applied Regression Analysis. John Wiley and Sons, New York.

Geoffrion, A. (1967). Integer programming by implicit enumeration and Balas' method. SIAM Review 9, 178-190.

Harter, H. L. (1972). The method of least squares and some alternatives. ARL Technical Report 72-0129, Aerospace Research Laboratories, Wright-Patterson Air Force Base, Ohio.

Kelley, J. E. (1958). An application of linear programming to curve fitting. SIAM Journal 6, 15-22.

LaMotte, L. R. and Hocking, R. R. (1970). Computational efficiency in the selection of regression variables. Technometrics 12, 98-93.

Narula, S. C. and Wellington, J. F. (1977). Subset selection with minimum sum of weighted absolute errors. Technical Report No. 37-77-P10, School of Management, Rensselaer Polytechnic Institute, Troy, New York.

Rice, J. R. and White, J. S. (1964). Norms for smoothing and estimation. SIAM Review 6, 243-256.

Roodman, G. (1974). A procedure for optimal stepwise MSAE regression analysis. Operations Research 22, 393-399.

THE SWEEP OPERATOR: ITS IMPORTANCE IN STATISTICAL COMPUTING

J. H. Goodnight
SAS Institute

Abstract

The importance of the Sweep Operator in Statistical Computing is not so much that it is an inversion technique, but rather that it is a conceptual tool for understanding the least squares process itself. The sweep operator can be programmed to produce generalized inverses, and create as by-products such items as: the Forward Doolittle matrix, the Cholesky Decomposition matrix, the Hermite canonical form matrix, the determinant of the original matrix, Type I Sums of Squares, the error sum of squares, a solution to the normal equations, and the general form of estimable functions. This tutorial paper begins by describing the use of Gaussian elemination for least squares and builds up to the description of a completely generalized sweep operator which computes and stores $(X'X)^-$, $(X'X)^-X'X$, $(X'X)^-X'Y$ and $Y'Y - Y'X(X'X)^-X'Y$ all in the space of a single upper triangular matrix.

1. Introduction

The Sweep Operator is perhaps the most versitile of all statistical operators. It can be adapted for use in ordinary least squares (including multiple regression and the general linear model), two and three stage least squares, non-linear least squares, multivariate analysis of variance, all possible regressions, regression by leaps and bounds, stepwise regression, partial correlation, and others. The most important aspect of the sweep operator with respect to its use as an instructional tool, is that each element of the matrix being operated on is readily identifiable and has statistical meaning. In order to understand the significance of each element of a matrix being swept and to understand the interrelationship between Gaussian elimination, the Forward Doolittle, Cholesky decomposition, Hermite canonical forms and the sweep operator, this paper will start at the most basic level, that of Gaussian elimination.

Throughout the paper the general linear model;

$$Y = X\beta + e \qquad (1)$$

will be used, where Y is an nxl vector of individual observations; X is an nxk matrix of 0's and 1's and/or continuous variables; β is a kxl vector of constant but unknown parameters, and e is distributed $NID(0,\sigma^2)$.

2. Gaussian Elimination

Gaussian elimination may be used to solve directly the normal equations:

$$X'X\,b = X'Y \qquad (2)$$

which arise from (1). Gaussian elimination involves only two operations, that of multiplying equations by a constant and that of adding a multiple of one equation to another. The following simple example illustrates these points.

Given the system:

$$4b_1 + 2b_2 = 6$$

$$2b_1 + 6b_2 = 10$$

Multiply the first equation by 1/4

$$b_1 + 1/2\ b_2 = 3/2$$

$$2b_1 + 6b_2 = 10$$

Next: Add -2* eqn. 1 to eqn. 2

$$b_1 + 1/2\ b_2 = 3/2$$

$$0b_1 + 5b_2 = 7$$

Next multiply eqn. 2 by 1/5

$$b_1 + 1/2\ b_2 = 3/2$$

$$0b_1 + b_2 = 7/5$$

Finally: Add -1/2 * eqn. 2 to eqn. 1

$$b_1 + 0b_2 = 4/5$$

$$0b_1 + b_2 = 7/5$$

This same sequence of operations can be carried out in a simple matrix tableau using equivalent row operations (multiplying a row by a constant and adding a multiple of one row to another. For example:

Given the tableau

$$\left[\begin{array}{cc|c} 4 & 2 & 6 \\ 2 & 6 & 10 \end{array}\right]$$

Multiply row 1 by 1/4

$$\left[\begin{array}{cc|c} 1 & 1/2 & 3/2 \\ 2 & 6 & 10 \end{array}\right]$$

Next: Add -2 * row 1 to row 2

$$\left[\begin{array}{cc|c} 1 & 1/2 & 3/2 \\ 0 & 5 & 7 \end{array}\right]$$

Next: Multiply row 2 by 1/5

$$\left[\begin{array}{cc|c} 1 & 1/2 & 3/2 \\ 0 & 1 & 7/5 \end{array}\right]$$

Finally: Add -1/2 * row 2 to row 1

$$\left[\begin{array}{cc|c} 1 & 0 & 4/5 \\ 0 & 1 & 7/5 \end{array}\right]$$

Thus to solve full rank regression equations one has only to form the augmented matrix

$$\left[X'X \mid X'Y\right] \qquad (3)$$

and use row operations to convert the left hand matrix to an identity. Once done, the right hand matrix will be the solution. Symbolically:

$$\left[X'X \mid X'Y\right] \xrightarrow[\text{operations}]{\text{row}} \left[I \mid b\right]. \qquad (4)$$

Note that $(X'X)^{-1}$ need not be computed in order to solve the normal equations, and that the entire process may be carried out in-place on a computer.

One of the most important aspects of performing row operations on a matrix is that it is equivalent to multiplying the matrix on the left by another matrix. Thus the row operations performed in (4) are equivalent to multiplying (3) by $(X'X)^{-1}$.

In addition to computing the b values, the error sum of squares may also be simultaneously computed by augmenting (3) as follows:

$$\left[\begin{array}{c|c} X'X & X'Y \\ \hline Y'X & Y'Y \end{array}\right]. \qquad (5)$$

If (5) is multiplied on the left by

$$\left[\begin{array}{c|c} (X'X)^{-1} & 0 \\ \hline -Y'X(X'X)^{-1} & I \end{array}\right] \qquad (6)$$

then the resultant product is:

$$\left[\begin{array}{c|c} I & (X'X)^{-1}X'Y \\ \hline 0 & Y'Y-Y'X(X'X)^{-1}X'Y \end{array}\right]. \qquad (7)$$

Since (7) was achieved by multiplying (5) on the left by matrix (6), (7) may also be achieved by performing row operations on (5). That is,

$$\left[\begin{array}{c|c} X'X & X'Y \\ \hline Y'X & Y'Y \end{array}\right] \xrightarrow[\text{operations}]{\text{row}} \left[\begin{array}{c|c} I & (X'X)^{-1}X'Y \\ \hline 0 & Y'Y-Y'X(X'X)^{-1}X'Y \end{array}\right] \qquad (8)$$

Since X'X is positive definite, the row operations used in (4) and (8), see Wilkinson (1961), need consist only of a sequence of pivots on the diagonal elements of X'X. In fact, by restricting the row operations to pivots on the diagonal elements of X'X, much valuable statistical information may be obtained, and an easily programmed algorithm is achieved. Since pivoting on a diagonal element "adjusts" the remaining matrix elements for the variable associated with the diagonal element, it is helpful to define this operation as an operator called the ADJUST operator.

Definition: ADJUST(K) operator

The ADJUST(K) operator performs a pivot on element a_{KK} of a matrix A as follows:

Step 1: Set B = a_{KK}

Step 2: Divide row K by B

Step 3: For each other row $i \neq k$
Set B = a_{iK} and subtract B*Row k from Row i.

Using the ADJUST operator on a simple example shows the amount of statistical information that is available through its use. For the regression model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + e$$

the augmented normal equations (5) would be:

$$A = \begin{bmatrix} N & \Sigma X_1 & \Sigma X_2 & \Sigma Y \\ \Sigma X_1 & \Sigma X_1^2 & \Sigma X_1X_2 & \Sigma X_1Y \\ \Sigma X_2 & \Sigma X_1X_2 & \Sigma X_2^2 & \Sigma X_2Y \\ \Sigma Y & \Sigma X_1Y & \Sigma X_2Y & \Sigma Y^2 \end{bmatrix}$$

If an ADJUST(1) operation if performed on A then:

$$A_1 = \left[\begin{array}{c|ccc} 1 & \bar{X}_1 & \bar{X}_2 & \bar{Y} \\ \hline 0 & \Sigma x_1^2 & \Sigma x_1x_2 & \Sigma x_1y \\ 0 & \Sigma x_1x_2 & \Sigma x_2^2 & \Sigma x_2y \\ 0 & \Sigma x_1y & \Sigma x_2y & \Sigma y^2 \end{array}\right]$$

The lower right 3x3 in A_1 is the corrected SS&CP matrix. The upper right 1x3 matrix consists of b values for the submodels: $X_1 = \beta_0' + e'$, $X_2 = \beta_0'' + e''$, and $Y = \beta_0''' + e'''$. The diagonals in the lower right 3x3 correspond to the error sum of squares for these submodels. If an ADJUST(2) operation is performed on A_1 then the following matrix results:

$$A_2 = \left[\begin{array}{cc|cc} 1 & 0 & b_0' & b_0'' \\ 0 & 1 & b_1' & b_1'' \\ \hline 0 & 0 & \Sigma x_2^2.1 & \Sigma x_2y.1 \\ 0 & 0 & \Sigma x_2y.1 & \Sigma y^2.1 \end{array}\right]$$

In A_2 the upper right 2x2 matrix contains the b values for the submodels: $X2 = \beta_0' + \beta_1' X_1 + e'$ and $Y = \beta_0'' + \beta_1'' X_1 + e''$. The lower right 2x2 matrix is the error SS&CP matrix for these two models. If an ADJUST (3) operation is performed on A_2 then the following matrix results:

$$A_3 = \left[\begin{array}{ccc|c} 1 & 0 & 0 & b_0 \\ 0 & 1 & 0 & b_1 \\ 0 & 0 & 1 & b_2 \\ \hline 0 & 0 & 0 & \Sigma y^2.1,2 \end{array}\right]$$

This last adjustment completes the solution of the normal equations and yields the error sum of squares in the bottom right position.

The Type I SS for each variable, see Barr et al. (1976), is an important by product of the ADJUST operator. If the augmented matrix A as in (5) is pxp, then whenever ADJUST(K) is made, the Y'Y element a_{pp} is reduced by $(a_{Kp})^2/a_{KK}$. Thus R(X_K | the other variables adjusted for) = $(a_{Kp})^2/a_{KK}$. An auxiliary vector can be employed to store the Type I SS prior to each use of ADJUST.

As noted above, after each use of ADJUST, the diagonals not yet ADJUSTED represent error sums of squares for submodels among the independent variables. Suppose that a matrix A as in (5) has been adjusted for 1,2,....,K-1. Then a_{kk} is the error SS for the model

$$X_k = \beta_0 + \beta_1 X_1 + \ldots \beta_{k-1} X_{k-1} + e.$$

The R^2 value for this model is then:

$$R_k^2 = (CSS_k - a_{kk})/CSS_k,$$

where CSS_k is the corrected SS for X_k. The R_k^2 value provides an invaluable singularity check. If $R_k^2 > .9999$ (for instance), then X_k is "statistically" at least, a linear combination of the preceeding X's. The elements in column k associated with previously adjusted variables spell out what the linear combination is. Recent articles by Frane (1977) and Berk (1977) examine further the use of the tolerance, $T = 1-R^2$, for checking against singularities. The tolerance T, will be discussed further in a later section concerning generalized inverses.

In the past few years, hundreds of articles have appeared concerning the numerical accuracy and stability of regression solutions obtained using Gaussian elimination. Beaton et al. (1976) and (1977) reference some of the more important articles and conclude that in many cases the attempt at estimating regression coefficients in highly collinear problems cannot be justified statistically. When data is highly collinear, then the technician in the lab who records the data may have more effect on the solution than does the technique used to achieve it. One of the most obvious facts about accuracy, that is often overlooked, is that when Gaussian elimination is used the mantissa length on the machine must at least be able to contain the largest element of X'X. Any program written in single precision is obviously headed for trouble when more than four significant digits appear in the input data. It is hoped that no least squares programs in common use are written in single precision when Gaussian elimination is used. For the very large class of least squares problems which involve only X's with 0's and 1's and whose largest X'X element is equal to the number of observations, Gaussian elimination programmed in double precision is more than adequite with respect to accuracy, and cannot be topped in terms of storage or speed.

3. Other ADJUST Uses

For multivariate linear models, where several dependent variables have the same set of independent variables, the b values and the error SS&CP matrix may be obtained simultaneously; i.e.:

$$\left[\begin{array}{c|c} X'X & X'Y \\ \hline Y'X & Y'Y \end{array}\right] \xrightarrow[\text{Columns of } X'X]{\text{ADJUST on}}$$

$$\left[\begin{array}{c|c} I & (X'X)^{-1}X'Y \\ \hline 0 & Y'Y-Y'X(X'X)^{-1}X'Y \end{array}\right] \qquad (9)$$

This is equivalent to the univariate case, except that Y has dimensions N x p instead of N x 1.

Partial correlation coefficients for two sets of variables, X_1(N x p) and X_2(N x q), may be obtained using the adjust operator. Suppose the correlations among the X_2's are needed after their adjustment for X_1, then proceed as follows (assume X_1 contains a colums of 1's):

$$\left[\begin{array}{c|c} X_1'X_1 & X_1'X_2 \\ \hline X_2'X_1 & X_2'X_2 \end{array}\right] \xrightarrow[\text{Columns of } X_1'X_1]{\text{ADJUST for}}$$

$$\left[\begin{array}{c|c} I & (X_1'X_1)^{-1}X_1'X_2 \\ \hline 0 & X_2'X_2-X_2'X_1(X_1'X_1)^{-1}X_1'X_2 \end{array}\right] \qquad (10)$$

Next, divide each element of $X_2'X_2 - X_2'X_1(X_1'X_1)^{-1}X_1'X_2$ by the square root of the two associated diagonals, and the partial correlation matrix is complete.

4. The Forward Doolittle

The basic Doolittle method, see Steel and Torrie (1960), was introduced in 1878 while Doolittle was an engineer with the U. S. Coast and Geodetic Survey. It was perhaps the first attempt to structure Gaussian elimination to fit the needs of the statistician. As it has evolved it is now referred to as the abbreviated Doolittle and consists of two parts. The first part, the Forward Doolittle, maps the sum of squares and cross products matrix into an upper triangular matrix. The second part, the Backward solution, computes the regression coefficients and the inverse matrix. Symbolically, the Forward Doolittle is used to map any symmetric positive definite matrix C into an upper triangular matrix A:

$$C = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \xrightarrow{\text{row operations}}$$

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} \\ 0 & C_{22.1} & C_{23.1} \\ 0 & 0 & C_{33.12} \end{bmatrix} = A$$

The Forward Doolittle matrix is constructed by adding multiples of the first row of the matrix to all following rows such that zeros are introduced below the diagonal in column one. Multiples of row two are then added to all following rows to introduce zeros below the diagonal in column two, etc. If the positive definiteness of the matrix is in question the tolerance check discussed in section three may be used. Note that if the ADJUST operator had been applied to the C matrix, and if the adjustments were made sequentially (k = 1,2,3,...), then prior to the adjustment for row k, row k of C is equal to row k of A.

When only the adjusted SS&CP matrix is needed as in (9) and (10), the Forward Doolittle affords a faster method of computing than the ADJUST operator. A further example of this is the computation of:

$$SS(H0: L\beta=0) = (Lb)' (L(X'X)^{-}L')^{-1} (Lb),$$

where L is estimable and b is any solution to (1). If the following matrix is formed:

$$\left[\begin{array}{c|c} L(X'X)^{-}L' & Lb \\ \hline (Lb)' & 0 \end{array}\right]$$

and a Forward Doolittle is performed on the diagonals of the upper left matrix, then the lower right matrix is transformed into:

$$0 - (Lb)' (L(X'X)^{-}L')^{-1} (Lb),$$

the negative of which is SS(Ho: $L\beta$ =0).

5. Factoring a Symmetric Positive Definite Matrix

If the symmetric positive definite matrix C has been mapped as follows:

$$C \xrightarrow[\text{Doolittle}]{\text{Forward}} A$$

then if each row of A is divided by its diagonal to produce a matrix B, then

$$B'A = C.$$

Furthermore, if each row of A is divided by the square root of its diagonal to produce a matrix U, then

$$U'U = C$$

The matrix U is called the Cholesky decomposition of C, see Isaacson and Keller (1966).

6. The Determinant of a Symmetric Positive Definite Matrix

After mapped C into A by the Forward Doolittle and created B by dividing each row of A by its diagonal, then:

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 0 & 0 \\ b_{12} & 1 & 0 \\ b_{13} & b_{23} & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{bmatrix}$$

$$\text{i.e.} \quad C = B'A$$

$$\text{therefore} \quad |C| = |B'|\ |A|.$$

$$= 1 \cdot \prod_i a_{ii}.$$

The determinant of an upper or lower triangular matrix is always the product of its diagonal elements, by the basic permuted products summation definition. Therefore the determinant is easily a by-product of both the Forward Doolittle and the ADJUST operator.

7. Eigen Values of $E^{-1}H$

In multivariate analysis, the eigen values and vectors of $E^{-1}H$ are needed, where E and H are symmetric positive definite matrices. Routines for eigen values abound for symmetric positive definite matrices, but $E^{-1}H$ is not symmetric positive definite. Using the mapping

$$E \xrightarrow[\text{Doolittle}]{\text{Forward}} A \longrightarrow U = \text{Cholesky decomposition}$$

and letting $L = U'$, then

$$(E^{-1}H - \lambda I)X = 0$$

$$(H - \lambda E)X = 0 \qquad \text{(multiply by E)}$$

$$(H - \lambda LU)X = 0 \qquad (E = U'U = LU)$$

$$(L^{-1}H - \lambda U)X = 0 \qquad \text{(multiply by } L^{-1}\text{)}$$

$$(L^{-1}HU^{-1} - \lambda I)UX = 0 \qquad \text{(factor out U).}$$

Since $L^{-1} = (U^{-1})'$, $L^{-1}HU^{-1}$ is a symmetric positive definite matrix. The eigen values of $E^{-1}H$ are the same as the eigen values of $L^{-1}HU^{-1}$. The eigen vectors of $E^{-1}H = U^{-1} *$ eigen vectors of $L^{-1}HU^{-1}$.

8. Finding the INVERSE

The inverse of a symmetric positive definite matrix A, may be found by augmenting an identity matrix to the right of A and then using the ADJUST operator on the diagonals of A. Symbolically:

$$[A \mid I] \xrightarrow[\text{diagonals}]{\text{ADJUST A}} [I \mid A^{-1}] \qquad (11)$$

Since repetitive use of the ADJUST operator is equivalent to multiplication on the left by some matrix, (11) is equivalent to (12).

$$A^{-1}\,[A \mid I] = [I \mid A^{-1}] \qquad (12)$$

Since in (11) the original identity matrix keeps count of the row operations being performed, this technique has been referred to as the method of counters.

The above process is also reversable since,

$$A[I \mid A^{-1}] = [A \mid I]$$

thus

$$[I \mid A^{-1}] \xrightarrow[\text{diagonals}]{\text{ADJUST } A^{-1}} [A \mid I]$$

9. Obtaining b Values, Error SS, and Inverse

By forming the following augmented matrix and adjusting the columns associated with X'X, the regression coefficients, error SS and inverse may be computed simultaneously, i.e.

$$\left[\begin{array}{c|c|c} X'X & X'Y & I \\ \hline Y'X & Y'Y & 0 \end{array}\right] \xrightarrow[X'X \text{ columns}]{\text{Adjust}}$$

$$\left[\begin{array}{c|c|c} I & b & (X'X)^{-1} \\ \hline 0 & \text{ESS} & -b' \end{array}\right]$$

This is equivalent to the matrix multiplication,

$$\left[\begin{array}{c|c} (X'X)^{-1} & 0 \\ \hline -Y'X(X'X)^{-1} & I \end{array}\right] \left[\begin{array}{c|c|c} X'X & X'Y & I \\ \hline Y'X & Y'Y & 0 \end{array}\right] =$$

$$\left[\begin{array}{c|c|c} I & b & (X'X)^{-1} \\ \hline 0 & \text{ESS} & -b' \end{array}\right]$$

Also note that

$$\left[\begin{array}{c|c|c} I & b & (X'X)^{-1} \\ \hline 0 & ESS & -b' \end{array}\right] \xrightarrow[(X'X)^{-1}\ \text{columns}]{\text{readjust}}$$

$$\left[\begin{array}{c|c|c} X'X & X'Y & I \\ \hline Y'X & Y'Y & 0 \end{array}\right]$$

The reversibility of adjustments allows one to ADJUST any of the original columns of X'X and thus "enter" that variable into the model, and also to readjust any column in the $(X'X)^{-1}$ area and thus "remove" that variable from the model. For example, consider the following data:

X_0	X_1	X_2	Y
1	1	1	1
1	2	1	3
1	3	1	3
1	1	-1	2
1	2	-1	2
1	3	-1	1

Form $\left[\begin{array}{c|c|c} X'X & X'Y & I \\ \hline Y'X & Y'Y & 0 \end{array}\right]$

	X_0	X_1	X_2	b	X_0	X_1	X_2	
STEP 0	6	12	0	12	1	0	0	
	12	28	0	25	0	1	0	
FORM MATRIX	0	0	6	1	0	0	1	(13)
	12	25	2	28	0	0	0	

STEP 1	1	2	0	2	1/6	0	0	
	0	4	0	1	-2	1	0	
ENTER X_0	0	0	6	2	0	0	1	(14)
	0	1	2	4	-2	0	0	

	X_0	X_1	X_2	b	X_0	X_1	X_2	
STEP 2	1	0	0	3/2	7/6	-1/2	0	
	0	1	0	1/4	-1/2	1/4	0	
ENTER X_1	0	0	6	2	0	0	1	(15)
	0	0	2	15/4	-3/2	-1/4	0	

STEP 3	1	0	0	3/2	7/6	-1/2	0	
	0	1	0	1/4	-1/2	1/4	0	
ENTER X_2	0	0	1	1/3	0	0	1/6	(16)
	0	0	0	37/12	-3/2	-1/4	-1/3	

STEP 4	1	2	0	2	1/6	0	0	
	0	4	0	1	-2	1	0	
REMOVE X_1	0	0	1	1/3	0	0	1/6	(17)
	0	1	0	10/3	-2	0	-1/3	

STEP 5	1	2	0	2	1/6	0	0	
	0	4	0	1	-2	1	0	
REMOVE X_2	0	0	6	2	0	0	1	(18)
	0	1	2	4	-2	0	0	

STEP 6	6	12	0	12	1	0	0	
	12	28	0	25	0	1	0	
REMOVE X_0	0	0	6	2	0	0	1	(19)
	12	25	2	28	0	0	0	

The above tableaus represent the following models with respect to Y.

Step	Model	b values	error SS
0	$Y = 0$		28
1	$Y = b_0$	2	4
2	$Y = b_0+b_1X_1$	3/2,1/4	15/4
3	$Y = b_0+b_1X_1+b_2X_2$	3/2, 1/4, 1/3	37/12
4	$Y = b_0+b_2X_2$	2, 1/3	10/3
5	$Y = b_0$	2	4
6	$Y = 0$		28

Observing the tableau at Step 3 (the full model), note that if X_1 is removed, the ESS will increase by $(1/4 * 1/4)/1/4 = 1/4 =$ Type II SS due to X_1. If X_2 is removed from the full model, the ESS will increase by $(1/3 . 1/3)/1/6 = 2/3 =$ Type II SS due to X_2. The Type II SS due to X_i (given the full model) is always $(b_i)^2/C_{ii}$, where C_{ii} is i^{th} diagonal element of $(X'X)^{-1}$. The Type II SS corresponds to the SS due to the hypothesis that $\beta_i = 0$.

10. Stepwise Regression

Stepwise regression (although usually employing the Sweep Operator) is illustrated by the following tableau. Suppose the original tableau has been formed and several adjustments made, i.e.

$$\left[\begin{array}{c|c|c} X'X & X'Y & I \\ \hline Y'X & Y'Y & 0 \end{array}\right] \xrightarrow[\text{Adjustments}]{\text{Several}} \left[\begin{array}{c|c|c} A & B & C \\ \hline D & E & F \end{array}\right]$$

(original tableau) (current tableau)

For any variable not yet entered, the error SS would be reduced by b_i^2/a_{ii} if it were entered. For any variable in the current model, the error SS would increase by b_i^2/c_{ii} if it were removed. The entrance SS (b_i^2/a_{ii}) and the exit SS (b_i^2/c_{ii}) form the basis for deciding which variable is to enter and which is to be removed at each stage.

11. The SWEEP Operator

The SWEEP operator is a modification of the ADJUST operator which reduces the amount of core needed to compute the b values, error SS, and $(X'X)^{-1}$. Whereas the ADJUST operator performs the in place mapping:

$$\left[\begin{array}{c|c|c} X'X & X'Y & I \\ \hline Y'X & Y'Y & 0 \end{array}\right] \xrightarrow[X'X \text{ Columns}]{\text{ADJUST}} \left[\begin{array}{c|c|c} I & b & (X'X)^{-1} \\ \hline 0 & \text{ESS} & -b' \end{array}\right],$$

the sweep operator performs the in place mapping:

$$\left[\begin{array}{c|c} X'X & X'Y \\ \hline Y'X & Y'Y \end{array}\right] \xrightarrow[X'X \text{ Columns}]{\text{SWEEP}} \left[\begin{array}{c|c} (X'X)^{-1} & b \\ \hline -b' & \text{ESS} \end{array}\right]$$

One of the first references to sweep operations may be found in Ralston (1960), but the term "Sweep Operator" was coined by Beaton (1964).

By observing tableaus (13)-(19), note that whenever a variable is in the model its associated X'X column is an identity column, and whenever a variable is not in the model its associated column in the original identity matrix remains unchanged. The SWEEP operator takes advantage of these features, and after adjusting the tableau for a particular variable, replaces the identity column just created by the associated column in the original identity matrix which has now been modified. By observing what happens to the i^{th} identity column when variable X_i is entered, the ADJUST operator can be modified to a SWEEP operator. When variable X_k is entered, row k is multiplied by $1/a_{kk}$, therefore its identity column will have $1/a_{kk}$ in the k^{th} row.

To zero out the remaining elements in column k, $-a_{ik}$ * row k is added to all other rows ($i \neq k$). Since the identity column has 0's in all positions other than row k, the i^{th} row of the identity column becomes $-a_{ik}/a_{kk}$ ($i \neq k$). With this in mind, the SWEEP operator is defined as follows.

SWEEP(K) defined:

Given an originally symmetric positive definite matrix A, SWEEP(K) will modify the current matrix A in the following manner:

Step 1: Let $D = a_{kk}$.

Step 2: Divide row k by D.

Step 3: For every other row $i \neq k$, let $B = a_{ik}$, Subtract B * row k from row i, then Set $a_{ik} = -B/D$.

Step 4: Set $a_{kk} = 1/D$.

Since the SWEEP operator is only a modification of the ADJUST operator, the reversibility of SWEEP is apparent. A useful exercise, for the interested reader, would be to use the SWEEP operator on the left most 4x4 matrix in tableau (13), to compute the equivalent sweep tableaus associated with (14)-(19).

It should be noted that, although the amount of core needed to compute the b values, error SS, and $(X'X)^{-1}$ has been reduced by using SWEEP, none of the by-product information has been lost. The Type I and II SS may still be computed. The determinant of X'X may still be computed. The singularity check is still available, and all submodel information is present. For example, let

$$A = \left[\begin{array}{c|c|c} X_1'X_1 & X_1'X_2 & X_1'Y \\ \hline X_2'X_1 & X_2'X_2 & X_2'Y \\ \hline Y'X_1 & Y'X_2 & Y'Y \end{array}\right]$$

After sweeping A on the columns associated with $X_1'X_1$,

$$\left[\begin{array}{c|c|c} (X_1'X_1)^{-1} & (X_1'X_1)^{-1}X_1'X_2 & (X_1'X_1)^{-1}X_1'Y \\ \hline -X_2'X_1(X_1'X_1)^{-1} & X_2'M_1X_2 & X_2'M_1Y \\ \hline -Y'X_1(X_1'X_1)^{-1} & Y'M_1X_2 & Y'M_1Y \end{array}\right] \quad (20)$$

where $M_1 = I - X_1(X_1'X_1)^{-1}X_1'$.

The regression coefficients for the submodels: Model $X_2 = X_1$; and Model $Y = X_1$; are clearly available. $X_2'M_1X_2$ is the error SSCP for the first model, and $Y'M_1Y$ is the error SSCP for the second model.

The ADJUST operator and the Forward Doolittle operator are all easily programmed to operate only on the upper triangular matrix by observing their symmetry properties. The SWEEP operator is also easily modified to operate on just the upper triangular portion of the matrix.

The amount of time it takes to compute $(X'X)^{-1}$, b values, and error SS using the SWEEP operator is generally only a fraction of the time it takes to form the sum of squares and cross product matrix. If only the upper triangular portion of the sum of squares and cross product matrix is formed as the data is being read, then one SWEEP operation (modified to operate on the upper triangle) takes about the same amount of CPU time as does the reading and accumulation of two observations. This can be verified by counting that number of multiplications and additions that are performed. Thus the approximate CPU time ratio of inversion to building the X'X is approximately 2k/N where k is the total number of independent variables and N is the number of observations. Since N is usually much larger than k, inversion represents only a fraction of the cost of regression analysis.

12. Computing Generalized Inverses

A large family of models, characterized by (1), are over-parameterized. Linear dependencies are known to exist among the columns of the X matrix. For some models, the dependencies are easily predictable. For others, the interchanging of effects (putting interactions before main effects), the absence of a cell, or the inclusion of a covariable may bring about unexpected dependencies. The ADJUST operator, the Forward Doolittle, and the SWEEP operator, through use of the tolerance check, can detect dependencies among the columns of the X matrix. In this section, the SWEEP operator will be modified to produce a generalized inverse and the corresponding particular solution whenever a dependency is encountered.

Initially assume that the X matrix can be partitioned, i.e., $X = [X_1 \mid X_2]$, such that the columns of X_1 are linearly independent, and each column of X_2 is a linear combination of the columns of X_1. This implies that:

$X_2 = X_1L$ (by definition),

$L = (X_1'X_1)^{-1}X_1'X_2$ (regress X_2 on X_1),

$X_2'X_2 - X_2'X_1(X_1'X_1)^{-1}X_1'X_2 = 0$ (error SS&CP).

The X'X matrix, $\begin{bmatrix} X_1'X_1 & X_1'X_2 \\ X_2'X_1 & X_2'X_2 \end{bmatrix}$ has two particular generalized inverses which are easily constructed through use of the SWEEP operator. The first, a g_1 inverse ($AA^{g_1}A = A$) for X'X, is

$$\left[\begin{array}{c|c} (X_1'X_1)^{-1} & (X_1'X_1)^{-1}X_1'X_2 \\ \hline -X_2'X_1(X_1'X_1)^{-1} & 0 \end{array}\right] .$$

This matrix results from SWEEPing X'X on the columns associated with $X_1'X_1$. If the full augmented tableau,

$$\left[\begin{array}{cc|c} X_1'X_1 & X_1'X_2 & X_1'Y \\ X_2'X_1 & X_2'X_2 & X_2'Y \\ \hline Y'X_1 & Y'X_2 & Y'Y \end{array}\right] \quad (21)$$

is swept on the columns associated with $X_1'X_1$, the resultant tableau is:

$$\left[\begin{array}{cc|c} (X_1'X_1)^{-1} & (X_1'X_1)^{-1}X_1'X_2 & (X_1'X_1)^{-1}X_1'Y \\ -X_2'X_1(X_1'X_1)^{-1} & 0 & 0 \\ \hline -Y'X_1(X_1'X_1)^{-1} & 0 & Y'Y-Y'X_1(X_1'X_1)^{-1}X_1'Y \end{array}\right] \quad (22)$$

Thus sweeping on the columns of $X_1'X_1$ yields a g_1 inverse and the appropriate adjusted Y'Y. However, the b values, $b_1 = (X_1'X_1)^{-1}X_1'Y$ and $b_2 = 0$, do not correspond to the b values computed using the g_1 inverse, i.e.

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} (X_1'X_1)^{-1} & (X_1'X_1)^{-1}X_1'X_2 \\ -X_2'X_1(X_1'X_1)^{-1} & 0 \end{bmatrix} \begin{bmatrix} X_1'Y \\ X_2'Y \end{bmatrix} .$$

$$b_1 = (X_1'X_1)^{-1}X_1'Y + (X_1'X_1)^{-1}X_1'X_2X_2'Y$$

$$= (X_1'X_1)^{-1}X_1'Y + LX_2'Y.$$

$$b_2 = -X_2'X_1(X_1'X_1)^{-1}X_1'Y$$

$$= -L'X_1'Y$$

$$= -X_2'Y.$$

Although $b_1 = (X_1'Y_1)^{-1}X_1'Y$ and $b_2 = 0$ is a solution to the normal equations this particular solution does not equal $(X'X)^{g1}X'Y$.

Another disadvantage of computing a g_1 inverse is that additional computition is needed to compute the variance of the particular solution since:

$$\text{Var}((X'X)^{g1}X'Y) = (X'X)^{g1}\,X'X\,(X'X)^{g1}\,\sigma^2.$$

On the other hand, a g_2 inverse ($AA^{g2}A = A$, and $A^{g2}\,A\,A^{g2} = A^{g2}$) of X'X has the property that:

$$\text{Var}((X'X)^{g2}\,X'Y) = (X'X)^{g2}\,\sigma^2.$$

It is easily verified that

$$\begin{bmatrix} (X_1'X_1)^{-1} & 0 \\ 0 & 0 \end{bmatrix}$$

is a g_2 inverse of X'X. If the full augmented tableau (21) is swept on the columns associated with $X_1'X_1$, then the resultant tableau is (22).

Introducing 0's in the two submatrices adjoining $(X_1'X_1)^{-1}$ in (22) yields:

$$\left[\begin{array}{cc|c} (X_1'X_1)^{-1} & 0 & (X_1'X_1)^{-1}X_1'Y \\ 0 & 0 & 0 \\ \hline -Y'X_1(X_1'X_1)^{-1} & 0 & Y'Y-Y'X_1(X_1'X_1)^{-1}X_1'Y \end{array}\right] \quad (23)$$

This tableau contains $(X'X)^{g2}$ and the adjusted Y'Y. The b values $b_1 = (X_1'X_1)^{-1}X_1'Y$ and $b_2 = 0$ are equivalent to $(X'X)^{g2}X'Y$, and therefore $\text{Var}(b) = (X'X)^{g2}\sigma^2$. If the g_2 inverse of (23) is employed, then the expected value of b_1 and b_2 are:

$$E(b_1) = E((X_1'X_1)^{-1}X_1'Y) = \beta_1 + (X_1'X_1)^{-1}X_1'X_2\beta_2$$

$$E(b_2) = E(0) = 0\ .$$

If the submatrix containing $(X_1'X_1)^{-1}X_1'X_2$ is saved after sweeping on the columns of $X_1'X_1$, then this matrix may be used to compute all possible solutions to the normal equations since (for Z arbitrary):

$$b^* = (X'X)^{g2}X'Y + (I-(X'X)^{g2}X'X)Z$$

$$= \begin{bmatrix} (X_1'X_1)^{-1}X_1'Y \\ 0 \end{bmatrix} + \begin{bmatrix} \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} \\ -\begin{pmatrix} I & (X_1'X_1)^{-1}X_1'X_2 \\ 0 & 0 \end{pmatrix} \end{bmatrix} \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} \quad (24)$$

Therefore,

$$b_1^* = (X_1'X_1)^{-1}X_1'Y - (X_1'X_1)^{-1}X_1.'X_2Z_2 = b_1 - LZ_2$$

$$b_2^* = 0 + Z_2 = Z_2$$

The foregoing discussion was simplified by assuming that the X's were grouped with X_1 being of full column rank and $X_2 = X_1L$. Naturally, this seldom ever happens but, as it turns out, the SWEEP operator can be modified to handle the situation in which the linearly independent columns of X are not all grouped.

For model (1), if it were known which columns of X formed a linearly independent basis then an index matrix M (an identity matrix with columns permuted) could be formed to rearrange the X's, and M' could be used to rearrange the β's. In other words, since $MM' = I$;

$$Y = X\beta + e$$

$$= XI\beta + e$$

$$= XMM'\beta + e.$$

Letting $Z = XM$ and $\gamma = M'\beta$, then $Y = Z\gamma + e$, where $Z = [Z_1 \mid Z_2]$ with the columns of Z_1 linearly independent and $Z_2 = Z_1L$. Thus

$$(Z'Z)^{g2} = \begin{bmatrix} (Z_1'Z_1)^{-1} & 0 \\ 0 & 0 \end{bmatrix}$$

and $\hat{\gamma} = (Z'Z)^{g2}Z'Y$.

Since $(Z'Z)^{g2} = (M'X'XM)^{g2} = M'(X'X)^{g2}M$

Then $(X'X)^{g2} = M(Z'Z)^{g2}M'$ and $\hat{\beta} = M\,\hat{\gamma}$. (25)

In essence, the above rearrangement process selects a maximum rank subset of elements from X'X, inverts the resulting matrix, then puts the inverted subset of elements back into their original positions and zero's out the remaining elements to form $(X'X)^{g2}$. The SWEEP operator can achieve the same results without any rearrangement of rows and columns, simply by proceeding as normal, except when encountering a variable which it finds to be linearly dependent on

previously swept variables. When a dependency is encountered, say for variable K, the SWEEP operator should set the K^{th} row and column to zero and proceed. A g_2 inverse and corresponding solution will result. Note well however, that once a row and column have been zeroed, the generalized SWEEP is no longer reversable. The critical information which defines that variables dependency has been set to zero. With these points in mind, the G2SWEEP operator is defined as follows:

Given an originally symmetric positive definite or symmetric positive semi-definite matrix A as in (21) which has not been SWEPTon column K, G2SWEEP(K) will modify the current matrix A in the following manner:

Step 1: Let $D = a_{kk}$
If D is less than $DMIN_k$ then set row and column k to zero and stop at this step. Otherwise proceed.

Step 2: Divide row k by D.

Step 3: For every other row $i \neq k$, let $B = a_{ik}$, subtract B * row k from row i, then set $a_{ik} = -B/D$.

Step 4: Set $a_{kk} = 1/D$.

As stated above, the G2SWEEP operator should not be applied but once to any given column of A, (although the order is unimportant) since information needed may have been set to zero. In addition a_{KK} once swept may be legitimately less than $DMIN_k$.

The $DMIN_k$ values used in the G2SWEEP operator are functions of the TOLERANCE value a... either the corrected SS(CSS) or uncorrected SS(USS) for variable k. If no intercept is employed in the model then:

$$DMIN_k = \begin{cases} \text{Tolerance if USS} = 0 \\ \text{Tolerance} * \text{USS otherwise} \end{cases}$$

If an intercept is employed in the model then:

$$DMIN_k = \begin{cases} \text{Tolerance if } X_k \text{ does not vary or has} \\ \text{a computed CSS} < = 0. \\ \text{Tolerance} * \text{CSS otherwise} \end{cases}$$

In checking to see if X_k varies (has more than two distinct values) if the smallest and largest non zero absolute values of X_k are computed, this affords the opportunity to check the ratio of the largest to smallest non-zero absolute values of X_k. If the ratio is greater than 1E7 (or 1E8) then the chances are that the computed X'X matrix in double precision is incorrect to start with.

Establishing the value to use for tolerance is not a simple task, since it involves (as does the comparison of two means) compromising between a Type I and Type II error. The major difference between choosing a Tolerance and an α level is the lack of distributional theory. Too large a tolerance may declare variables dependent on others when they in fact are not. Too small a tolerance may let dependencies go unchecked. For variables which take on only 0 or 1 values, $(1-R^2)$ values in the range 1E-12 to 1E-14 are quite common when dependencies occur. Fifth and sixth degree polonomials may exhibit $(1-R^2)$ values as small as 1-E8 even when they are linearly independent. Thus Tolerance values in the range of 1E-8 to 1E-12 seem appropriate when looking for dependencies.

13. The Upper Triangular G2SWEEP

As discussed by Schatzoff, et al. (1968) the amount of time it takes to perform a sweep operation may be reduced by approximately one half by taking into account the symmetry properties of the sweep tableau. The amount of core needed to store the tableau may also be reduced by almost one half by saving and operating on only the upper triangular portion of the tableau. Since operating on the upper triangular tableau, stored in a one-dimension array, is an obvious extension of what is to be discussed here, the tableau will be assumed to be stored in a rectangular array, but reference will be made only to elements on or above the diagonal.

Starting with the following tableau:

$$\begin{bmatrix} X_1'X_1 & X_1'X_2 & X_1'Y \\ X_2'X_1 & X_2'X_2 & X_2'Y \\ Y'X_1 & Y'X_2 & Y'Y \end{bmatrix} \qquad (26)$$

perform a G2SWEEP on the $X_1'X_1$ columns. This results in:

$$\left[\begin{array}{c|c|c} (X_1'X_1)^- & (X_1'X_1)^-X_1'X_2 & (X_1'X_1)^-X_1'Y \\ \hline -X_2'X_1(X_1'X_1)^- & X_2'M_1X_2 & X_2'M_1Y \\ \hline -Y'X_1(X_1'X_1)^- & Y'M_1X_2 & Y'M_1Y \end{array}\right] \qquad (27)$$

where $(X_1'X_1)^-$ is a symmetric g_2 inverse and $M_1 = I - X_1(X_1'X_1)^-X_1'$. Note that this tableau is symmetric, except for the sign of the submatrices below $(X_1'X_1)^-$. Letting a_{ij} denote the elements of this tableau, then the elements below the diagonal a_{ij} (with $i > j$) may be constructed from the elements above the diagonal in the following manner:

$$a_{ij} = \begin{cases} a_{ji} & \text{if i and j have been swept.} \\ a_{ji} & \text{if neither i nor j has been swept.} \\ -a_{ji} & \text{if i or j (but not both) has been swept.} \end{cases}$$

By keeping track of which columns have been swept, the lower triangular portion of the tableau can be constructed from the upper.

The easiest way to keep track of what has been swept and what has not is through use of an auxiliary vector V. The elements of V are initially set to 1, then after sweeping on column K, V_k is set to $-V_k$. If this vector V is employed, then elements below the diagonal may be constructed from the upper triangular portion as follows:

$$a_{ij} = a_{ji} * V_i * V_j \quad (\text{for } i > j)$$

Note that the upper triangular elements are always correct as is. It is only when an element is not in the upper triangular portion of the tableau that multiplying a_{ji} by $V_i * V_j$ is necessary. The upper triangular G2SWEEP makes the same assumptions about the matrix A as does the G2SWEEP operator. Up to the point where a dependency is declared, reversability is possible. The vector V is assumed to have been set to 1 prior to any sweeps. UTG2SWEEP(K) modifies the upper triangular portion of the matrix A in the following manner:

Step 1: Let $D = a_{kk}$
If $V_k = 1$ and $D <= DMIN_k$ then zero elements above and to the right of a_{kk} including a_{kk}, then terminate.
Otherwise proceed.

Step 2: For each value of i: i = 1,2,...,NROWS except for i = k perform Step 3
Then go to step 5

Step 3: If i < k then $B = a_{ik}/D$
Otherwise $B = V_i * V_k * a_{ki}/D$
Then for each value of j = i, i+1,..., NCOLS except for j = k perform step 4

Step 4: If k < j then $c = a_{kj}$
Otherwise $c = V_j * V_k * a_{jk}$
Set $a_{ij} = a_{ij} - B*C$

Step 5: For each value of i: i = 1,...,k
set $a_{ij} = -a_{ij}/D$
For each value of j: j = k,...,NCOLS
set $a_{kj} = a_{kj}/D$
then set $a_{kk} = 1/D$ and $V_k = -V_k$

14. The Reversable Upper Triangular G2SWEEP

By modifying the UTG2SWEEP such that it does not zero the k^{th} row and column when a dependency occurs, reversability may be achieved. In addition the resulting tableau will contain both $(X'X)^{g2}$ and $(X'X)^{g2}X'X$ which are easily separated, along with $(X'X)^{g2}X'Y$ and $Y'Y - Y'X(X'X)^{g2}X'Y$. To demonstrate these properties, reconsider the case in which X was partitioned into [X1 ¦ X2] with X_1 of full column rank and $X_2 = X_1L$. Sweeping the initial tableau:

$$\begin{bmatrix} X_1'X_1 & X_1'X_2 & X_1'Y \\ X_2'X_1 & X_2'X_2 & X_2'Y \\ Y'X_1 & Y'X_2 & Y'Y \end{bmatrix} \qquad (28)$$

on the columns of $X_1'X_1$ yields:

$$\begin{bmatrix} (X_1'X_1)^{-1} & (X_1'X_1)^{-1}X_1'X_2 & (X_1'X_1)^{-1}X_1'Y \\ -X_2'X_1(X_1'X_1)^{-1} & 0 & 0 \\ -Y'X_1(X_1'X_1)^{-1} & 0 & Y'M_1Y \end{bmatrix} \qquad (29)$$

where $M_1 = I - X_1(X_1'X_1)^{-1}X_1'$. Sweeping (29) on the columns of $(X_1'X_1)^{-1}$ yields (28), therefore reversability can be achieved. Note also in (28) and (29) that the same symmetry properties discussed for the UTG2SWEEP also hold.

The $(X'X)^{g2}$ matrix can be obtained from the final tableau (in which no columns are left to sweep) by use of the V vector described earlier. Letting G represent the g_2 inverse, and A the final tableau, then

$$g_{ij} = \begin{cases} 0 & \text{if } V_i + V_j >= 0, \text{ otherwise} \\ a_{ij} & \text{if } i <= j \text{ or } a_{ji} \text{ if } i > j \end{cases}$$

Using the g_2 inverse described above,

$$(X'X)^{g2}X'X = \begin{bmatrix} I & (X_1'X_1)^{-1}X_1'X_2 \\ 0 & 0 \end{bmatrix}$$

Letting H represent $(X'X)^{g2}X'X$, and A the final tableau, then

$$h_{ii} = \begin{cases} 0 & \text{if } V_i = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$h_{ij} = \begin{cases} 0 & \text{if } V_i + V_j \neq 0, \text{ otherwise} \\ a_{ij} & \text{if } i <= j \text{ or } -a_{ji} \text{ if } i > j \end{cases}$$

As was discussed with the G2SWEEP operator it is not necessary that the columns of X be grouped, since the tolerance check can be used to determine which variables are linearly dependent on the ones previously swept. In defining the reversable UTG2SWEEP, the same initial tableau and V vector as used in the UTG2SWEEP will be assumed.

The RUTG2SWEEP(K) modifies the upper triangular portion of the matrix A in the following manner:

Step 1: Let $D = a_{kk}$
If $V_k = 1$ and $D <= DMIN_k$ then terminate. Column K cannot be swept at this time. Otherwise proceed.

Step 2 - Step 5 are the same as for UTG2SWEEP.

Using the RUTG2SWEEP, the Type II SS for any effect may be computed by starting with the tableau at any stage and making the appropriate sweeps.

15. Using RUTG2SWEEP Sequentially

By using the RUTG2SWEEP sequentially (K = 1,2,...,NX) all of the statistics described previously are available. Prior to any sweep, row k's diagonal and elements to the right of the diagonal correspond to the k'th row of the Forward Doolittle, thus $(a_{ky})^2/a_{kk}$ (provided $a_{kk} > DMIN_k$) is the Type I SS for variable k. Also the product of each diagonal (just prior to sweeping) equals the determinant of X'X. When sweeping is done sequentially then $(X'X)^{g2}X'X$ is the Hermite canonical form H, is upper triangular and the elements of H (except for some 0's and 1's) are already in place in the final tableau. H can also be computed by sequentially pivoting on each non-zero row of X'X. Since the expected value of the solution achieved using RUTG2SWEEP sequentially is equal Hß, the nature of the bias in the solution is at hand. The H matrix is also one of many matrices whose rows can be used as a generating set to compute any estimable function. In other words, any linear combination of the rows of H produce a matrix L such that Lß is estimable, see Goodnight (1976).

16. Summary

The SWEEP operator and its extensions are indeed versatile tools which not only afford solutions to the normal equations and a gammet of additional statistics but also allow complete insight into the nature of least squares. The general concepts of the SWEEP operator, once mastered, allow the whole least squares process to be visualized. Without this conceptual tool it is extremely difficut to explain such concepts as Absorbtion and what the R notation is testing in terms of the parameters of the model. With the SWEEP tool in mind those concepts are readily grasp.

References

(1976) Barr, A. J. et al. A User's Guide to SAS 76, SAS Institute, P. O. Box 10066, Raleigh, N. C. 27605

(1964) Beaton, A. E. The Use of Special Matrix Operators in Statistical Calculus, Educational Testing Service, RB-64-51.

(1976) Beaton, A. E., D. B. Rubin and J. L. Barone. The Acceptability of Regression Solutions: Another Look at Computational Accuracy JASA, March 1976.

(1977) _____. Comment on More on Computational Accuracy in Regression, JASA September 1977.

(1977) Berk, K. N. Tolerance and Condition in Regression Computations. JASA December 1977.

(1977) Frane, J. W. A Note on Checking Tolerance in Matrix Inversion and Regression. Technometrics, November 1977.

(1976) Goodnight, J. H. Computational Methods in General Linear Models, ASA Proceedings of the Statistical Computing Section, August 1976.

(1966) Isaacson, E. and H. B. Keller, Analysis of Numerical Methods, John Wiley and Sons, New York.

(1960) Ralston, Anthony, Mathematical Methods for Digital Computers, p. 191, Wiley, New York.

(1968) Schatzoff, M., R. Tsao, S. Fienberg Efficient Calculation of All Possible Regressions, Technometrics, November 1968.

(1960) Steel, R. G. D. and J. H. Torrie, Principles and Procedures of Statistics, McGraw-Hill, New York.

(1961) Wilkinson, J. H. Error analysis of direct methods of matrix inversion. J. Ass. Comp. Mach. #8, p. 281-330.

REJECTION USING PIECE-WISE LINEAR MAJORIZING FUNCTIONS IN RANDOM VARIATE GENERATION

Bruce W. Schmeiser
and
Mohamed A. Shalaby

Southern Methodist University

ABSTRACT

In the context of random variate generation on digital computers, the use of piece-wise linear majorizing functions in conjunction with the general rejection algorithm is proposed. Based on previous results obtained in the generation of beta variates, the expected advantages and disadvantages of applying the concept to other distributions are discussed, as is the use of minorizing functions for fast acceptance of values. Areas of potential application are also discussed.

I. INTRODUCTION

Much literature in the last twenty years has been devoted to process generation on digital computers. Process generation is the creation of a sequence of observations having the properties of some desired distribution or process. Almost always process generation is a transformation of one or more uniform (0,1) values to the desired distribution. Common methods for performing the transformation include using the inverse distribution function transformation, rectangular approximation, special properties, composition, and rejection. (See for example, [2,5,6].) In the past most interest has centered on the first three methods. Recently composition and rejection have received much attention. In a wide variety of cases, rejection is fast, easy to code, and requires little memory.

In this paper the use of piece-wise linear rejection functions and methods for fast acceptance of observations are discussed for the univariate continuous case. The beta distribution is used as an example, drawing on the results of Schmeiser and Shalaby [8]. Discussion centers on concepts necessary for generalizing the results to other distributions. The general rejection algorithm is discussed in Section II and specialized to the piece-wise linear case in Section III. Discussed are limitations in Section IV, fast acceptance in Section V, and potential applications in Section VI.

II. THE GENERAL REJECTION ALGORITHM

In this section a general form of random variate generation using rejection is given. This general form is specialized to the piece-wise linear special case. Implications and applicability of the piece-wise linear approach are discussed.

The common rectangular rejection algorithm can be generalized as follows. Let $p(x)$ be the density function from which random variates are to be generated. Let $t(x)$ be a majorizing function of $p(x)$; i.e., $t(x) \geq p(x)$ for all x. Corresponding to $t(x)$ is the density function $r(x) = t(x) / k$, where $k = \int_{-\infty}^{\infty} t(x)\,dx$. Figure 1 illustrates the relationship of $p(x)$, $t(x)$ and $r(x)$. The general rejection algorithm for generating variates from $p(x)$ is

1. Generate a value x from $r(\cdot)$.
2. Generate a value u from the rectangular distribution over the interval $[0,1]$.
3. If $u \leq p(x) / t(x)$, accept x by setting $y = x$. Otherwise go to step 1.

Proposition: The algorithm provides values of y from the distribution having density function $p(\cdot)$.

Proof: Let A denote the event that step 3 results in acceptance. In any given iteration

$$P(A \mid x) = p(x) / t(x) = p(x) / [k\, r(x)]$$

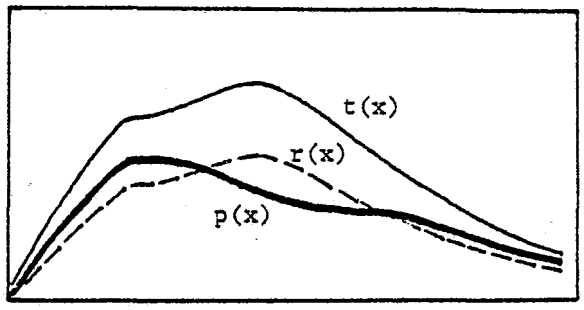


Figure 1. The functions used in the general rejection algorithm for generating random variates from p(x).

and therefore

$$P(A \mid X_\varepsilon I) = \frac{P(A \text{ and } X_\varepsilon I)}{P(X_\varepsilon I)}$$

$$= \frac{\int_I P(A \mid x)\, r(x)\, dx}{\int_I r(x)\, dx}$$

$$= \frac{\int_I [p(x)/t(x)]\, r(x)\, dx}{\int_I r(x)\, dx}$$

$$= \frac{\int_I p(x)\, dx}{\int_I t(x)\, dx}$$

and

$$P(A) = \int_{-\infty}^{\infty} P(A \mid x)\, r(x)\, dx = 1/k$$

Let Y be the random variable resulting from the algorithm. It is necessary to show that $P(Y_\varepsilon I) = \int_I p(x)\, dx$ for any interval I. The proof follows directly from

$$P(Y_\varepsilon I) = P(X_\varepsilon I \mid A)$$

$$= P(A \mid X_\varepsilon I)\, P(X_\varepsilon I) \,/\, P(A)$$

$$= \frac{\int_I p(x)\, dx}{\int_I t(x)\, dx} \int_I r(x)\, dx \,/\, [1/k]$$

$$= \int_I p(x)\, dx \qquad \text{as desired.} \blacksquare$$

Although stated in a different form, this rejection algorithm is mathematically equivalent to that described by Tocher [11, p. 25].

For a given majorizing function $t(x) = k\, r(x)$, k is selected to be as small as possible while still maintaining $t(x) \geq p(x)$ for all x. This results in maximizing the probability P(A) that in any given iteration the value x generated in step 1 will be accepted in step 3.

For a given density function p(x) the choice of majorizing function $t(x) = k\, r(x)$ plays a central role in determining whether or not the resulting algorithm is efficient. The majorizing function must both have nearly the same shape as p(x) (thereby resulting in a small value of k) and a density function r(x) which is amenable to variate generation (via any technique, but probably not rejection).

The reason for early disfavor of rejection was the selection of a uniform distribution for r(x). (In fact, many textbooks discuss only this special case.) The rectangular assumption restricts consideration to distributions having a finite range or to approximations obtained by truncation. While such approximations may be made as accurate as desired in theory, great inefficiency results from using a rectangular distribution to model tails having small probabilities.

In the last several years the use of non-rectangular rejection regions has appeared more frequently in the literature. The gamma distribution especially has been the topic of several papers [1,9,10,12]. All of these papers have used density functions r(x) corresponding to well-known distributions. The basic results of these papers is the identification of suitable functions r(x) and the determination of k such that valid and efficient algorithms result.

III. PIECE-WISE LINEAR MAJORIZING FUNCTIONS

While the use of common theoretical distributions for r(x) has been fruitful, another approach which is very general and often easy to apply is to use piece-wise linear majorizing functions. Piece-wise linearization calls for partitioning the range of the random variable into segments such that t(x) is linear over each segment. The usual rectangular rejection region is a special case corresponding to only one segment. Another special case, briefly discussed by Lewis [5, p. 82], is the use of "regular parts," which is a discontinuous piece-wise linear majorizing function having only rectangular segments. More generally, however, the linear segments may lie at the angle providing the best fit to p(x). As an example, Schmeiser and Shalaby [8] used a piece-wise linear majorizing function in considering rejection methods for the beta distribution. Figure 2 illustrates the algorithm for a particular beta density function.

Step 1 of the algorithm requires generation of variates from the density $r(x) = t(x)/k$. Now the piece-wise linear r(x) is composed of a mixture of rectangular, triangular, and trapezoidal densities. Note in Figure 2 that the trapezoidal densities are each composed of a rectangular lower density and a triangular upper density. Thus, $r(x) = \sum_{i=1}^{n} \alpha_i\, r_i(x)$ where $\sum_{i=1}^{n} \alpha_i = 1$ and $0 \leq \alpha_i \leq 1$ for all i, i = 1,2,..., n and each $r_i(x)$ is either a rectangular or a triangular density.

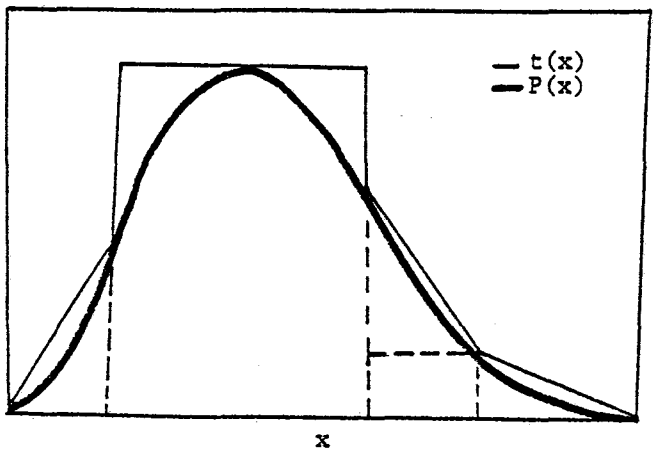


Figure 2. Rejection algorithm using a piece-wise linear majorizing function t(x).

The generation of variates from a piece-wise linear r(x) (using the composition method) requires the generation of a variate from $r_i(x)$ with probability a_i. The rectangular densities may be easily generated using $x = a + (b - a)\,u$ where u is a uniform (0,1) variate and a and b are the bounds of the rectangular density function. The triangular densities require $x = a + (b - a) \max(u_1, u_2)$ when $r_i(x)$ has a positive slope and $x = a + (b - a) \min(u_1, u_2)$ when $r_i(x)$ has a negative slope, where u_1 and u_2 are independently generated uniform (0, 1) variates. Since generation from a piece-wise linear r(x) requires no exponential level operations, step 1 can be executed quite rapidly.

In addition the probability of acceptance in step 3 can be made close to one, since a piece-wise linear majorizing function can be made to fit any density function p(x) arbitrarily well by increasing the number of segments. Here a trade-off developes between few segments resulting in simple coding (with associated minimal memory requirements) and many segments resulting in longer code, more memory requirements, but higher probability of acceptance. The use of even a few segments provides a considerably better fit than the simple rectangular region. For example, in reference [8] three and five segments are used in two of the beta generation algorithms. While the single segment provides an algorithm which is not competitive for many beta parameter values, five segments are the nucleus of the fastest algorithm available for many parameter values.

IV. LIMITATIONS

The applicability of the rejection technique is dependent only upon the selection of the majorizing function. For a particular density p(x), the minimal value of k such that $k\,r(x) \geq p(x)$ for all x is central to the applicability of the rejection technique. This inequality implies two conditions: 1) k r(x) must be greater than zero whenever p(x) is greater than zero and 2) $\lim_{x \to a} k\,r(x) = \infty$ whenever $\lim_{x \to a} p(x) = \infty$. Since k must be greater than one and finite these two conditions become 1) r(x) must be greater than zero whenever p(x) is greater than zero and 2) $\lim_{x \to a} r(x) = \infty$ whenever $\lim_{x \to a} p(x) = \infty$. Since the piece-wise linear majorizing function cannot be non-zero at infinity nor be infinite, the piece-wise approach is applicable only as an approximation to distributions having densities with one or more infinite values and to distributions with infinite length ranges.

These are two theoretically important restrictions. For example, the beta distribution with parameters less than one, the gamma distribution with shape parameter less than one, and some members of a general family of distributions of Schmeiser and Deutsch [7] have points at which p(x) is infinite. In addition, many distributions have infinite length ranges, most commonly $(-\infty, \infty)$ and $[0, \infty)$.

However these restrictions are not important in practice. First consider the problem of $\lim_{x \to a} p(x) = \infty$. Few computers have accuracy beyond 10^{-8}, nor do many applications require more accuracy. If ε is the minimum discernable accuracy, then an approximation using the finite values $r(a + \varepsilon)$ or $r(a - \varepsilon)$ rather than the infinite r(a) will probably be quite acceptable. The second problem of infinite length range may be overcome by including so much of the range that the excluded portion will never be missed. For example, consider the normal distribution. While having a range of $(-\infty, \infty)$, the probability of observing a point more than ten standard deviations from the mean is only 1.524 x 10E-23. Of course, if this is unacceptable then one hundred standard deviations can be included with little additional cost.

Although the theoretical limitations are not important, the use of a piece-wise linear majorizing function does require that the density function p(x) yields r(x) with a reasonable amount of effort. For distributions having only a single shape this is not important, since an appropriate r(x) may be determined once and for all. For example, see Kinderman and Ramage's [4] normal variate generator. However, families of distributions such as the gamma and beta include multiple shapes. A generator designed to generate values from any member of the family must be able to quickly determine an appropriate majorizing function. For example the beta generators of reference [8] use the equations for the location of the mode and points of inflexion to locate the junctures of the piece-wise linear segments. The points of inflexion are critical since the convexity or concavity of the density function at various points is necessary to prove that indeed $k\,r(x) \geq p(x)$ for all x.

V. FAST ACCEPTANCE

In addition to choosing r(x) such that the probability of acceptance is high, another function b(x) may be chosen to reduce the time necessary to determine whether or not $u \leq p(x) / t(x)$ in step 3. If b(x) is substantially faster to evaluate than p(x) and if $b(x) \leq p(x)$ for all x, then the algorithm may be made faster by replacing step 3 with

3(a). If $u\,t(x) \leq b(x)$, accept x by setting $y = x$.
3(b). If $u\,t(x) \leq p(x)$, accept x by setting $y = x$. Otherwise go to step 1.

The theory underlying the algorithm has not changed, since step 3(a) accepts x only if 3(b) would accept x anyway. However, the use of step 3(a) often makes the evaluation of p(x) unnecessary. Since density functions often include time consuming operations such as exponential and gamma functions, the savings due to this minorizing function can be substantial. In particular, a piece-wise linear minorizing function is often easy to determine and is always fast to evaluate.

Sometime a minorizing function is not necessary. If a trapezoidal region is formed by t(x) which is composed of triangular and rectangular regions, then the rectangular region is often entirely under p(x). In this case no check is necessary in step 3, the value of x being accepted automatically.

Note also that the minorizing function b(x) may be used with any majorizing function, not just the piece-wise linear functions discussed in Sections II and III.

VI. POTENTIAL APPLICATIONS

There are a number of distributions to which the above concepts can be applied. The gamma distribution generators currently available involve several logarithmic operations. The use of piece-wise linear majorizing and minorizing functions would almost certainly be faster. Pearson distributions other than the gamma may also be amenable to the piece-wise linear approach. Two families which have well-known, but slow, generators, the Weibull and lognormal, could also be generated using this approach. The F and t distributions, classically generated using their relationships to the normal or the beta distributions, could be more quickly generated using the above techniques. In addition, the J-shaped beta family could benefit from such techniques. (Jöhnk's algorithm [3] for U-shaped and the algorithms discussed in Schmeiser and Shalaby [8] for bell-shaped beta distributions probably preclude much faster times using the above techniques.)

It is also possible that the piece-wise linear techniques can be applied to discrete distributions such as the Poisson and binomial. Current generators for these two distributions require times proportional to the mean of the Poisson and to the number of trials for the binomial. The commonly used normal approximations could be avoided by the use of rejection techniques.

VII. SUMMARY AND CONCLUSIONS

The commonly used rectangular rejection region has been generalized and the most general form of rejection has been specialized to the use of piece-wise linear majorizing functions. The implications of the piece-wise linear approach have been discussed and potential applications have been mentioned.

The rejection algorithm using the piece-wise linear majorizing function has both advantages and disadvantages compared to other rejection methods. The disadvantage is that the determination of the piece-wise linear majorizing function can require a non-trivial set-up cost. The expected advantages are

1. applicability to a wide variety of distributions,
2. fast and easy generation of x in step 1,
3. high probability of acceptance in step 3, and
4. fast acceptance in step 3(a).

REFERENCES

1. Fishman, G.S. "Sampling from the gamma distribution on a computer," Communications of the ACM, 19, 7 (July 1976), 407-409.
2. Jannson, B. Random Number Generators, Almquist and Wiksell, Stockholm, 1966.
3. Jöhnk, M.D. "Erzeugung von Betaverteilten und Gammaverteilten Zufallszahlen," Metrika, 8 (1964), 5-15.
4. Kinderman, A.J. and Ramage, J.G. "Computer generation of normal random variables," Journal of the American Statistical Association, 71 (December 1971), 893-896.
5. Lewis, T.G. Distribution Sampling for Computer Simulation, D.C. Heath and Co., Lexington, Mass., 1975.
6. McGrath, E.J. and Irving, D.C. Techniques for Efficient Monte Carlo Simulation. Volume II. Random Number Generation for Selected Probability Distributions, NTIS, 5285 Port Royal Road, Springfield, VA 22151, 1973.
7. Schmeiser, B.W. and Deutsch, S.J. "A versatile four-parameter family of probability distributions, suitable for simulation," AIIE Transactions, 9, 2 (June 1977), 176-182.
8. Schmeiser, B.W. and Shalaby, M.A. "Rejection methods for beta variate generation," Technical Report IEOR77014, Southern Methodist University, Dallas, TX 75275.
9. Tadikamalla, P.R. "Computer generation of gamma random variables," Communications of the ACM, forthcoming.
10. Tadikamalla, P.R. "Computer generation of gamma random variables -- II," Technical Report, Department of Business Administration, Eastern Kentucky University, Richmond, KY 40475.
11. Tocher, K.D. The Art of Simulation, English Universities Press, London, 1963.
12. Wallace, N.D. "Computer generation of gamma random variates with non-integral shape parameters," Communications of the ACM, 17 (1974), 691-695.

IMPLEMENTATION OF HARMONIC DATA ANALYSIS PROCEDURES

Michael E. Tarter
Department of Biomedical and Environmental Health Sciences
University of California, Berkeley

ABSTRACT

An interactive and graphical data analysis system, GRAFSTAT, has been developed whose underlying structure is based on harmonic or Fourier series methods. The general organizational framework and content of the system is outlined in this paper.

0. Major concerns and constraints

One general goal underlay the development of GRAFSTAT. This was to make univariate and bivariate harmonic data analysis procedures usable. Certainly these new techniques could not deal with all data analysis applications. Therefore, a wide variety of conventional procedures such as histograms, scatter, fractile and frequency diagram options are included as program options alongside their harmonic nonparametric estimation counterparts.

The basic structure of GRAFSTAT can be divided into nine subsystems, one of which, VIEW/EDIT DATA, is central in the sense that almost all other subsystems are accessible when the VIEW/EDIT DATA frame is on the cathode ray tube, CRT, screen. The figure given below describes the distribution of the nine subsystems in terms of calling sequence. As denoted by the dual heads on all but one arrow, there is only one one-way street, i.e., a sequence from which data must be re-entered before accessing alternative sequences.

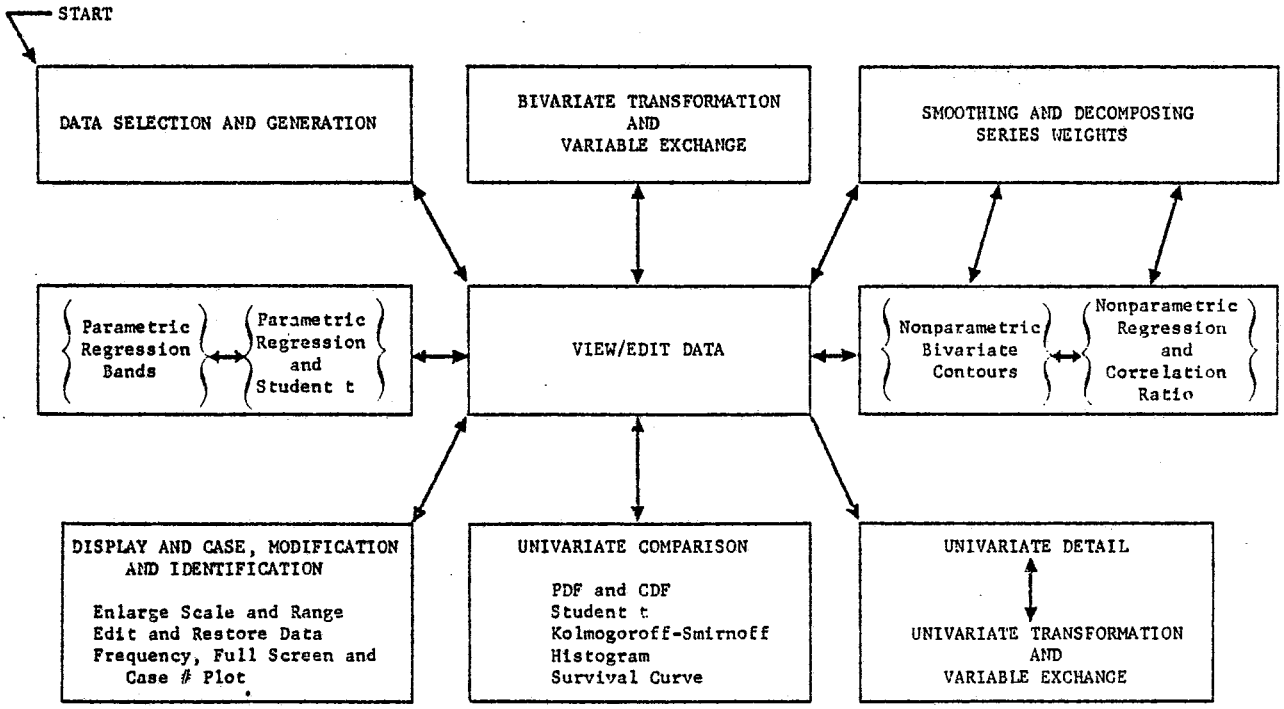


Figure 1. GRAFSTAT CALLING PATTERN

Research supported in part by National Institutes of Health, National Cancer Institute Grant 1-R01CA21448-01

Only two of the nine subsequences, DATA SELECTION AND GENERATION and LINEAR REGRESSION AND BANDS, do not contain options based on harmonic analysis. In most program subsequences, both harmonic and nonparametric estimation procedures are accessible alongside their conventional counterparts. This makes it fairly simple for a user to compare results obtained by new and old methods.

With only two exceptions, the label of any immediately accessible frame or option is always displayed on the CRT screen. The exceptions, accessed by CONTROL 0 or Z, allow a user to return to the immediately preceding frame or immediately exit the program. Unlike the highway where a driver finds him- or herself proceeding in a wrong direction, and where a long distance must be travelled before reversing direction, GRAFSTAT provides offramps for immediate change of direction whenever the user selects CONTROL 0.

In order to avoid cluttering the CRT face with unnecessary detail, only minimal scaling and other numerical information is automatically displayed. However, a user who wishes to obtain the numerical coordinate of any point on a displayed curve, scatter or other diagram can, by pointing with the light pen to the appropriate location, have the associated numerical values presented. Since GRAFSTAT allows for an unlimited degree of display enlargement, the roughness of light pen sensing can be overcome to provide any needed degree of accuracy (up to the degree of precision of our numerical procedures). When dealing with the variable "case number," a special option can be executed to find the exact position in the data file of any displayed point.

Special options can be accessed by a sequence of identical control selections. For example, the first display which appears after selecting the UNIVARIATE COMPARISON - HISTOGRAM option overlays the histograms constructed from two subgroups of data. To display the two histograms individually, one need only select HISTOGRAM a second or a third time.

All but 0 and Z options may be initiated by light pen sensing the option's label on the CRT screen. With only three exceptions all options can also be accessed by key-board. This form of selection is usually preferable to light-pen sensing. The light pen, however, plays an important role in the execution of many options. For example, to change the number of class intervals of a histogram in the UNIVARIATE COMPARISON segment, one need only light-pen sense on any symbol of the CRT display CLASS INTERVAL = 20. A cursor will then appear beneath the digit 2 and allow the user to change the number 20 by key-board entry.

Although details of our computing hardware and software as well as new statistical procedures cannot even be skimmed over in this brief paper, one very important comment seems mandatory before proceeding to outline the nine segments of GRAFSTAT. It would have been impossible to have developed this system without the IMGRAF language and considerable technical assistance provided by W. Dixon, J. Johnson, C. Newton and K. Ryder of the UCLA Health Sciences computing facility. Too many individuals at our end of the roughly four hundred miles which separates our IMLAC terminal from UCLA have contributed to GRAFSTAT to make it practical to even list their names. However, without the assistance of the HSCF Graphics group our own programmers and other scientists could not have completed and used a small portion of the procedures which are outlined below.

1. Data Selection and Generation

The first frame presented to a user allows the ability to choose between the selection of natural, and the generation of artificial, data. If a file of natural data had been analyzed prior to selection of the artificial data generation frame, initial values of artificial data parameters will be set equal to estimates obtained from the most recently analyzed file of natural data.

The natural data selection frames allow the user to easily enter subsets of data and measurements of up to six variables for further analysis. The size of the twenty-six natural data files which are currently available on our system ranges from ten cases, each containing two variables, to a maximum of three thousand cases and twenty nine variables.

2. View|Edit Data

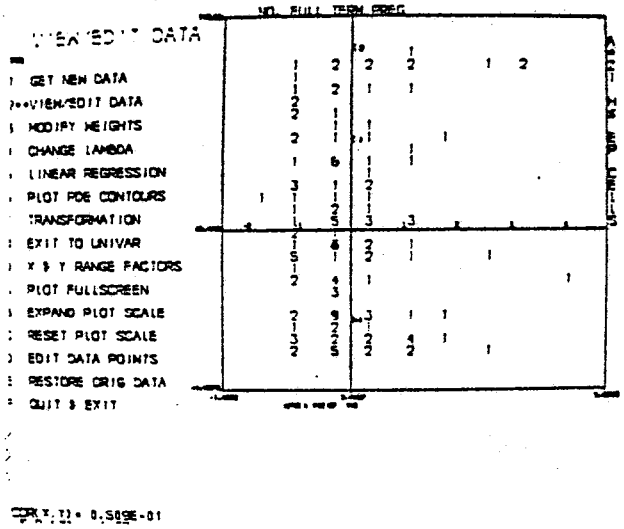


FIGURE 2. VIEW|EDIT DATA FRAME

The VIEW|EDIT DATA frame consists in part of a scatter or frequency diagram of variables designated as X and Y in the DATA SELECTION AND GENERATION segment. Coordinate axes are constructed to pass through the sample means $\bar{X}$ and $\bar{Y}$ while the axes are graduated in sample standard deviation units. The X and Y sample median pair is marked by a small triangle placed within the display. The original and post-editing sample sizes as well as numerical values of the sample correlation and X and Y standard deviations are projected on the screen.

To obtain the coordinates of an observation represented on the scatter diagram the user need only sense the appropriate point with the light pen. In certain applications, by plotting case number against a variable's numerical value and using the expand plot scale option, numerical values can be associated with individualized cases. A more easily utilized procedure for case identification, which lists point coordinates and associated file rank, is accessible whenever the EXPAND PLOT SCALE option has left ten or fewer points on the screen.

3. Bivariate Transformations and Exchange of Variables

Variables can be transformed, modified and exchanged using a variety of procedures, ranging

from a single log transformation to conventional regression as well as nonparametric-ordac residual calculation.

Although GRAFSTAT is designed primarily to perform bivariate and univariate analyses, an X or Y variable or both variables simultaneously can be modified or combined with up to four additional variables. A user can either create a new transformed variable, or replace an old variable with a new value. The latter process allows an unlimited series of transformations to be executed in the memory space provided for four Z or supplemental, as opposed to the display or XY, variables.

By exchanging a Z with an X or Y variable the user can view a variety of bivariate combinations without returning to the DATA SELECTION AND GENERATION FRAME 1. After any transformation or editing option has been executed, a RESTORE ORIGONAL DATA option is provided in the VIEW|EDIT DATA frame. By using this option it is possible to quickly undo the results of transformation and editing processes without returning to Frame 1.

4. Parametric Regression and Student's t

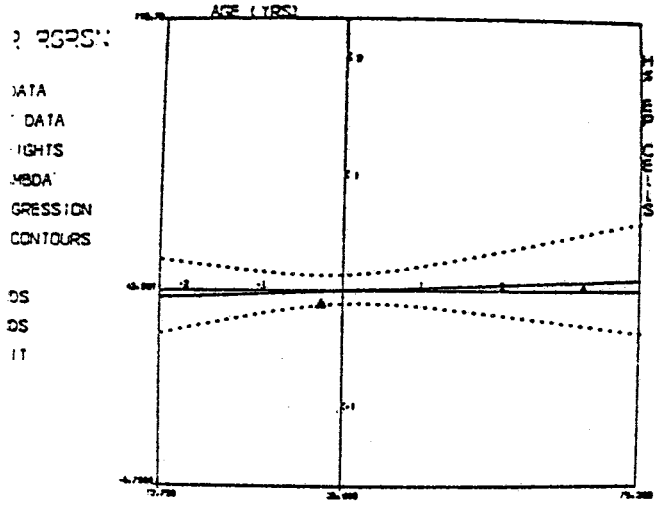


FIGURE 3. TYPICAL CONVENTIONAL LINEAR REGRESSION AND BANDS

The bivariate option, LINEAR REGRESSION, allows a user to obtain a least squares line fitted to the points displayed in the VIEW|EDIT DATA frame. After selecting LINEAR REGRESSION, a user can elect to obtain a set of 95% confidence or prediction bands for this line.

The equation for the least squares line, an estimate of error $\sigma_{y|x}$ and the Student's t statistic for the null hypothesis that regression slope equals $\beta=0$ are also displayed. By choosing a dichotomous variable as X and a normally distributed variable as Y, a user can use the LINEAR REGRESSION frame to obtain a Student's t test for group difference.

5. Nonparametric Bivariate Contours, Regression and Correlation

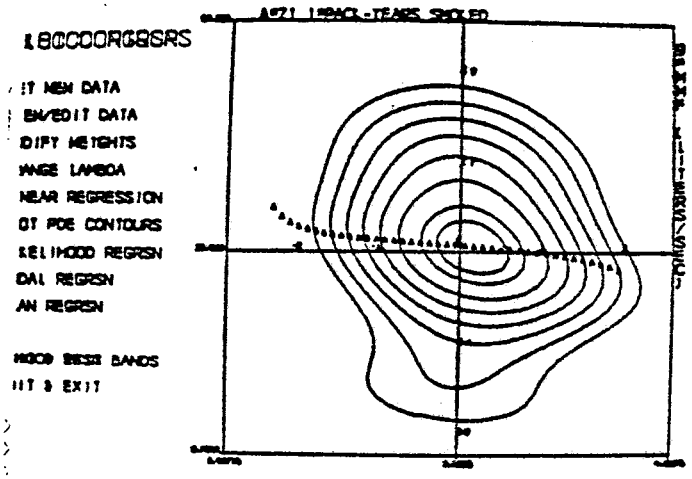


FIGURE 4. PDE CONTOURS AND NONPARAMETRIC REGRESSION CURVE

By selecting option, PLOT PDE CONTOURS, a user can obtain isopleths of a generalized histogram constructed above the X,Y variate plane. The "terraces" of this estimated "hill" of probability have several uses among which are the following:

A. The separation of contour subsets implies bivariate bimodality and the existence of distinct population subcomponents.

B. The slant and length of the major axis of an ellipse fitted to a contour tends to indicate degree of relationship (see Tarter and Silvers (1975) Section 4).

C. By light-pen sensing on members of a list of contour heights provided on the lower left hand portion of the frame, the user can erase or reconstruct any of the displayed contours. This option provides a "feel" for the three dimensional structure of frequency associated X and Y variable values.

D. The size of the lowest contours in relationship to the axes origin, i.e., the sample means and graduations, i.e., the sample standard deviations, conveys information about bivariate skewness and kurtosis.

A variety of nonparametric regression options are accessible from PLOT PDE CONTOURS. The coordinates of any point on a nonparametric regression curve can be obtained by light-pen sensing. By sensing on two points and obtaining their coordinates, a user can compare the position of nonparametric and parametric regression curves.

Three procedures can be executed from Frame 5 which utilize nonparametric estimates of the standard deviation of the Y variable for a given value of the X variable, i.e., estimates of $\hat{\sigma}_{y|x}$. The most useful of these options displays an estimate of the correlation ratio $\sqrt{1-\sigma^2_{y|x}/\sigma^2_y}$. (Note if the X,Y variables are bivariate normal $\sqrt{1-\sigma^2_{y|x}/\sigma^2_y} = \rho$, the population correlation coefficient.) In a sense the

correlation ratio provides a local as opposed to global estimate of variate association and bivariate non-normality (see Rietz (1924), pp. 129-31).

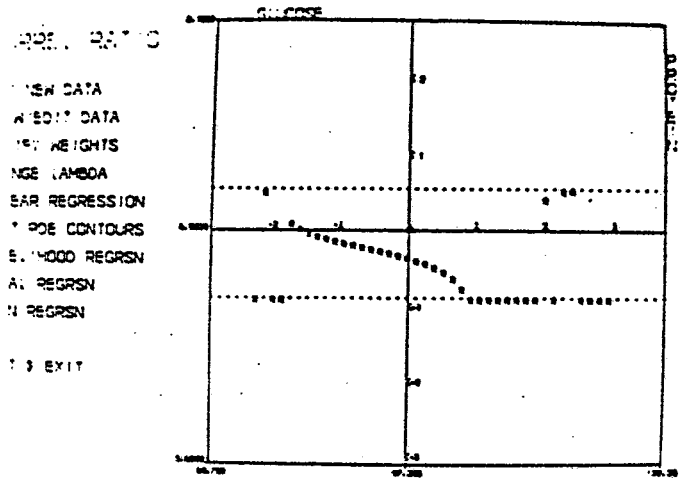


FIGURE 5. CORRELATION RATIO ESTIMATOR

The correlation ratio estimate shown in Figure 5 indicates that for the GLUCOSE-PROTEIN BLOOD LEVELS data being considered in this example, small values of blood glucose levels tend to be more closely associated with differences in blood protein levels than do large values. An alternative interpretation of this figure is that variation in protein levels is much greater for individuals with high glucose blood levels than it is for individuals with low glucose blood levels.

6. Univariate Comparison

By selection of the option labeled UNIVAR(Y), a user can compare the univariate distributions of two data subgroups by means of a variety of displays. The first frame presented after UNIVAR(Y) is chosen is a modification of VIEW|EDIT data designed to allow a user to select appropriate data subgroups. Typically, the X or key variable of this VIEW|EDIT modification will be an attribute or dichotomous variable, e.g., SEX = 0 implies male and SEX = 1 implies female. The program allows up to four separate intervals to be selected within which values of the X or key variable will assign cases to the T or test group.

After specifying the intervals upon which the test group will be based, the program will display cumulative distribution function estimates for the test and its complement group. A Kolmogoroff-Smirnoff, K-S, test statistic will be approximated as well as the asymptotic 95% K-S critical point appropriate for the sample sizes of the test and test complement group (see Siegel (1956), p. 131).

By selecting PROB. DENSITY, a user can obtain a pair of estimated densities associated with the above cumulatives. Since the standard group difference t-tests can be obtained from the parametric regression option, the univariate comparison frame displays a "t-test" statistic which is computed very differently from its parametric regression counterpart. Roughly speaking, local as opposed to global estimation procedures are used to estimate the parameters of the two subpopulations. Preliminary research seems to indicate that for some applications these local estimators tend to be more insensitive to outliers than are conventional estimators.

By selecting HISTOGRAM, a user can obtain a pair of histograms for the groups of data being compared in the UNIVARIATE COMPARISON ROUTINE. An additional histogram option is provided in the VIEW|EDIT segment. The difference between these options is that the UNIVARIATE COMPARISON variant adjusts the areas under each of the two histograms to be equal. This is not done by the VIEW|EDIT histogram segment. Thus if one wishes to compare proportions, use of the UNIVARIATE COMPARISON histograms is indicated. If group sample sizes differ, one can graphically compare actual frequencies or counts, rather than proportions, i.e., counts scaled in accordance with sample size, by the VIEW|EDIT data histogram.

After selecting the UNIVARIATE COMPARISON HISTOGRAM OPTION, histograms associated with both data groups will be presented for purposes of comparison. A second and then a third sensing on HISTOGRAM allows a user to view each of the histograms individually.

The UNIVARIATE COMPARISON histogram segment allows a user to specify a choice of from 1 to 30 class intervals. The VIEW|EDIT data histogram, which is designed to correspond to the VIEW|EDIT - PLOT FULLSCREEN frequency diagram option, uses 22 class intervals which is the number of classes associated with the frequency diagram option.

The UNIVARIATE COMPARISON segment contains a survival curve estimation option. This program section has been designed to link to our clinical life-table type options in order to utilize combinations of complete and incomplete data. The same procedure used to select data subsets for comparison purposes is used by the survival curve estimation option to distinguish complete from incomplete observations.

7. Univariate Detail

As indicated by its name, the basic GRAFSTAT program BHLNK is in reality a combination of bivariate and univariate subprograms. The bivariate portion of BHLNK tends to contain many more conventional procedures, e.g., data editing and residual construction options, than does the univariate portion UNIVAR. Conversely, UNIVAR, either as automatically linked to the bivariate section of BHLNK or else as used alone, contains many new nonparametric estimation procedures which are not available in its bivariate counterpart. For example, although there are few transformation options included in UNIVAR, those that are included contain an "automated" suggested transformation constant section (see Tarter and Kowalski, 1972). Specifically, although a user can use the transformation Y' = log(Y-C)in both programs, in UNIVAR a "best" value for the constant C will be suggested while in the bivariate transformation segment, the user is given no assistance in the selection of C. (Note that setting C=0 can lead to serious computational as well as statistical problems for certain samples of data.) He can of course copy down the value of C provided by UNIVAR for later use in other segments of BHLNK.

Several display modification and enlargement options are available in UNIVAR. The user who wishes to edit his data is advised to perform editing

operations before branching to UNIVAR. However, while UNIVAR permits only a few operations to be performed on data, there are a large number of options available in this program for operations involving separation and isolation of population subcomponents (see Tarter, et al., 1976).

For example, the long and short term components of cancer survival probability densities can be separated, and then compared or analyzed individually.

. A large number of parameter estimation alternatives and transformation effectiveness checks are provided in UNIVAR. Besides conventional sample mean and median as well as estimated population median and mode, UNIVAR permits a user to estimate location and, in some instances, scale by a variety of new procedures (see Tarter 1975, a and b). These methods can be applied after population subcomponents have been isolated.

Individualized density, cumulative or survival curve components can be reassembled by a UNIVAR option. This procedure can both provide a check on the decomposition process and in some cases provide improved estimates due to the use of different density estimation constants for different population subcomponents.

A variety of UNIVAR options are designed to counteract side effects of either kernel or series density estimation procedures. For example, the option REDUCE first estimates and then compensates for excess variance of displays based on non-negative kernels. The option PERTURB tends to remove the distorting effects of Fourier series periodicity.

Several UNIVAR options are designed to overcome problems associated with particular types of data or underlying distributions, e.g., densities with straggling tails. We have found our method for group correction (the nonparametric analog of Sheppard's correction) useful in a variety of cancer and other studies.

8. Smoothing and Decomposing Series Weights

The methods described in the last section are all designed to deal with specific and known data analysis obstacles, e.g., clumping or data collected with minimal precision. These methods do not necessarily smooth a display and in particular, the REDUCE option, which corrects for non-negative kernel induced excess variance, tends to roughen a display (see Tarter and Raman, 1972).

There are many instances where a user may wish to obtain a smoothed display. High contrast may not be desirable since minor details may obscure important global distributional features (see Tarter and Kronmal, 1976, Section 2). For this reason, the procedures previously described for superimposed component variance reduction have been designed to work in reverse and allow a user to smooth most distributional displays.

Two distinct types of smoothing procedures are actually available in all BHLNK subsegments. The first procedure, the λ method, is based upon properties of the cumulant generating function (see Kronmal, 1964 and Tarter and Silvers, 1975) and the second FEJER WEIGHTS, is based on series summation techniques (see Kronmal and Tarter, 1968). The former procedure can be used either to blur or resolve detail, i.e., decrease or increase contrast. The latter can only be used to decrease contrast. The above methods can be used simultaneously. The FEJER WEIGHT method has in a few instances tended to temper a display so that the λ method can more effectively resolve previously hidden detail.

FEJER WEIGHTS and the λ methods are available in both UNIVAR and other segments of the program. The λ method tends to be very useful when used in conjunction with the CONTOUR program segments since in the bivariate case covariance modification often is an effective procedure for teasing apart distributional subcomponents (see Tarter and Silvers (1975) Section 3).

9. Display and Case Modification and Identification

Many options have been provided to modify displays and edit data. Axis scale as well as the margins between the data range and harmonic estimation period can be set automatically or manually. When a series of displays are used for comparison purposes, e.g., by using thermofax transparency overlays, manual input of identical axis scales allows displays to be more easily compared. All data editing and display modification procedures can be reversed.

Since the VIEW|EDIT data frame lists a large number of options, we have provided a procedure which utilizes the space otherwise reserved for the option list to provide an enlarged display. By choosing PLOT FULLSCREEN, one can also reset the display point threshold which automatically determines whether a scatter or frequency diagram will be presented. For example, before modification by the user, this point threshold is at sample size n = 720. At this setting, if more than 720 points are to be displayed, counts of points occurring within preset intervals rather than individual points will be presented. This option both protects the Cathode Ray Tube Phosphor from being damaged by the projection of a large number of coincident points, and permits a user to obtain a better grasp of data distributions over rich as opposed to sparse regions of concentration.

When fewer than ten points are projected on the screen, an option, SHOW CORRESP CASES, is made available which lists the point coordinates together with the order of the points in the file. This option was designed for outlier detection follow-up purposes. The SHOW CORRESP CASES option can be used in conjunction with the EXPAND PLOT SCALE option to find the file rank and X,Y coordinates of any data point subset.

REFERENCES

Cencov, N. N., "Evaluation of an unknown distribution density from observations," Soviet Mathematics, 3 (1962) 1559-1562.

Davis, K. B., "Mean square error properties of density estimates," The Annals of Statistics, 3 (July 1975) 1025-1030.

Kronmal, R. A., "The estimation of probability densities, unpublished doctoral dissertation, Department of Public Health, University of California, Los Angeles, CA, 1964.

Kronmal, R. A. and Tarter, M. E., "The estimation of probability density and cumulatives by Fourier series methods," Journal of the American Statistical Association, 63 (September 1968) 925-952.

Rietz, H. D., ed., HANDBOOK OF MATHEMATICAL STATISTICS, Boston: Houghton Mifflin Company, 1924.

Siegel, S. NONPARAMETRIC STATISTICS FOR THE BEHAVIORAL SCIENCES, New York: McGraw-Hill, 1956.

Tarter, M. E., "Model-free data exploration and pattern recognition -- Discussion and examples," Proceedings of Computer Science and Statistics: Eighth Annual Symposium on the Interface, February, 1975a, 372-381.

Tarter, M. E., "A method for maximum likelihood parameter estimation based upon a reduced data file," Proceedings of the Conference on Computer Graphics, Pattern Recognition and Data Structure, Institute of Electrical and Electronics Engineers, IEEE Catalog Number 75CH0981-1C, 1975b, 249-252.

Tarter, M. E. and Kowalski, C. J., "A new test for, and class of transformations to, normality," Technometrics, 14 (August 1972) 735-743.

Tarter, M. E. and Kronmal, R. A., "On multivariate density estimates based on orthogonal expansions," Annals of Mathematical Statistics, 41 (April 1970) 718-722.

Tarter, M. E. and Kronmal, R. A., "An introduction to the implementation and theory of nonparametric density estimation," The American Statistician, 30, No. 3 (August 1976) 105-112.

Tarter, M. E., Marshall, J. S., Lum, S. B., Rigsbee, E. O., Wong, J. T., "Interactive editing of biomedical data," Computer Programs in Biomedicine, 6 (July 1976) 117-123.

Tarter, M. E. and Raman, S., "A systematic approach to graphical methods in biometry," Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 4, Berkeley and Los Angeles: University of California Press, 1972, 192-221.

Tarter, M. E. and Silvers, A., "Implementation and applications of bivariate Gaussian mixture decomposition," Journal of the American Statistical Association, 70 (March 1975) 47-55.

Tarter, M. E., "Towards the harmonic analysis of data," (in press).

Tukey, J. W., "The technical tools of statistics," The American Statistician, 19 (April 1965) 23-28.

Watson, G. S., "Density estimation by orthogonal series," Annals of Mathematical Statistics, 40 (August 1969) 1496-1498.

FINDING INFLUENTIAL SUBSETS OF DATA IN REGRESSION MODELS

Roy E. Welsch and Stephen C. Peters

Center for Computational Research
Sloan School of Management
Massachusetts Institute of Technology

Abstract

This paper discusses a variety of techniques for identifying influential subsets of data in estimated regression models. Single and multiple row methods using deletion and infinitesimal perturbations are treated. The primary emphasis is on the algorithmic issues arising in the computation of these diagnostic measures.

1. Introduction and Notation

In this paper we explore a number of ways to identify subsets of data that appear to have a disproportionate influence on estimated linear regression models and to diagnose which parts of the estimated model are most affected by these subsets. Our main goal is to discuss some of the algorithms needed to compute these diagnostic measures and to point out some unsolved problems. A more detailed discussion of many of these diagnostic measures is contained in Welsch and Kuh (1977) and Hoaglin and Welsch (1978).

The regression model will be denoted by $y = X\beta+\varepsilon$ with X an $n\times p$ matrix and the least-squares estimates for β by b. Parentheses, as in b(i), will be used to denote what row or subset of rows has been deleted from the computation. The estimated error variance will be called s^2 and $r_i = y_i - x_i b$.

2. Single Row Methods

We begin by presenting techniques for discovering influential observations which examine each row separately. An influential observation is one which, when perturbed, has a relatively larger impact on the estimated coefficients, standard errors, etc. than is the case for most of the other observations. One obvious means for finding such observations is to delete each row, one at a time, and note the resultant effect on the various calculated values.

Since the estimated coefficients are often of primary interest, we look first at the estimated coefficients, b(i), obtained by deleting the i-th row. The change caused by this deletion is given by

$$\text{DFBETA} \equiv b-b(i) = \frac{(X^TX)^{-1}x_i^T r_i}{1-h_i} \qquad (1)$$

where

$$h_i = x_i(X^TX)^{-1}x_i^T . \qquad (2)$$

In order to assess the relative magnitude of this change for the j-th component, b_j, we divide by an estimate of the standard error of b_j to obtain

$$\text{DFBETAS}_{ij} \equiv \frac{b_j-b_j(i)}{s(i)\sqrt{(X^TX)^{-1}_{jj}}} \qquad (3)$$

where we have replaced the usual estimate of σ^2, s^2, by

$$s^2(i) = \frac{1}{n-p-1}\sum_{k\neq i}[y_k-x_k b(i)]^2$$

in order to make the denominator stochastically independent of the numerator in the Gaussian case. A simple formula for s(i) is

$$(n-p-1)s^2(i) = (n-p)s^2 - \frac{r_i^2}{1-h_i}. \qquad (4)$$

The Euclidean norm of DFBETAS for each i is one way to summarize all of the coefficient changes for a given row. Another way, which does not depend on the particular coordinate system used to form the regression model, is the scaled change in fit

$$\text{DFFITS}_i = \frac{x_i(b-b(i))}{s(i)\sqrt{h_i}} = \left(\frac{h_i}{1-h_i}\right)^{\frac{1}{2}}\frac{r_i}{s(i)\sqrt{1-h_i}} . \qquad (5)$$

We often multiply this by (n-p)/p to remove some of the dependence on n and p. Additional details are contained in Welsch and Kuh (1977).

We can see from (1) to (5) that h_i and r_i are fundamental components of deletion diagnostics. Some properties of h_i will be discussed in the remainder of this section and we will study special types of residuals (like $r_i/s(i)\sqrt{1-h_i}$) in the next section.

The h_i are the diagonal elements of the least-squares projection matrix, also called the hat matrix,

$$H = X(X^TX)^{-1}X^T \tag{6}$$

which determines the fit

$$\hat{y} = Xb = Hy. \tag{7}$$

A detailed discussion of H is contained in Hoaglin and Welsch (1978).

The influence of the response value, y_i, on the fit is most directly reflected in its leverage on the corresponding fitted value $\hat{y}_i$, and this information is seen from (7) to be contained in h_i. The h_i may also be interpreted as distance measures. Let $\tilde{X}$ denote the X matrix after centering by removing the column means. Then $h_i = \tilde{h}_i + 1/n$ and

$$\tilde{h}_i = \tilde{x}_i(\tilde{X}^T\tilde{X})^{-1}\tilde{x}_i^T \tag{8}$$

which is one form of distance from x_i to $\bar{x}$. The Mahalanobis distance from x_i to the center of all other observations, $\bar{x}(i)$, is

$$M_i^2 = (n-2)(\tilde{x}_i-\bar{\tilde{x}}(i))(\tilde{X}^T(i)\tilde{X}(i))^{-1}(\tilde{x}_i-\bar{\tilde{x}}(i))^T \tag{9}$$

$$= \frac{n(n-2)}{n-1}\cdot\frac{\tilde{h}_i}{(1-1/n)-h_i}$$

and guidelines about what constitutes a large value for h_i can be derived from M_i^2.

We now turn to the computation of some of these measures when X is of full rank and well conditioned. All of these computations are based on the orthogonal decomposition

$$X = QR \tag{10}$$

where Q is n×p with $Q^TQ = I$ and R is p×p upper triangular. The least-squares coefficients are found stably and efficiently by back substitution in the triangular system

$$Q^Ty = Rb . \tag{11}$$

Q and R can be computed most reliably by using Householder transformations (Golub, 1965). An often used alternative, Modified Gram-Schmidt (MGS) does not suit our purposes because, without additional reorthogonalization steps, the Q delivered by MGS can be severely nonorthogonal; Householder transformations do not have this defect. We require orthogonal Q because if $Q^TQ = I$ then $H = QQ^T$. The cost of determining the least-squares solution by Householder transformations is approximately $np^2-p^3/3$ operations where an operation is one floating-point multiply and one floating-point addition. When X is ill-conditioned more work is required. A rather complete treatment is given in Golub, Klema, and Stewart (1977).

Although in later sections we will need some of the off-diagonal elements of $H = QQ^T$, we will focus here on the diagonal elements which are given by

$$h_i = \sum_{j=1}^{p} q_{ij}^2 . \tag{12}$$

We can compute the h_i in parallel by forming in turn each column of Q, squaring its n elements, and adding these new squares to the previous totals.

The columns of Q are found by applying the elementary symmetric orthogonal transformations $H_1, H_2, \ldots, H_p$ which determine Q to p columns of the n×n identity matrix. The cost of forming the h_i in this way is about $2np^2$ operations.

Finally we consider the computation of the b-b(i) given in (1). The residuals are formed directly and b-b(i) for i=1,...,n are obtained from

$$B = (X^TX)^{-1}X^TA = R^{-1}Q^TA \tag{13}$$

where A is a diagonal matrix with elements $r_i/(1-h_i)$. The columns of B are found by backsolving the system

$$RB = Q^TA . \tag{14}$$

Given h_i and Q, B is formed in about $np^2/2$ additional operations. Note that care must be exercised in forming the quotient $r_i/(1-h_i)$ when h_i is very near one, but this occurrence is seldom found in practice.

3. Subset Methods

If we just look at each row separately, the influence of one point may be masked by another or the true impact and nature of a group of influential observations may not be fully diagnosed. This implies that we need to examine sets of rows of size larger than one. In this section we will discuss four approaches based on dummy variables, derivatives, covariance measures, and geometry. Each of these approaches can be specialized to single rows and used to supplement the basic measures discussed in the previous section.

3.1 Dummy Variables

A number of authors (Mickey, Dunn, and Clark, 1967 and Wood, 1973) have suggested adding dummy variables, d_i, with a one in row i and zeros elsewhere to the data matrix, X, in order to assess which observations might not be well represented by the regression model. Usually a base set, B*, of potentially misrepresented rows is chosen by some means (such as partial residual plots) or by the one row at a time methods (with relaxed cut-off levels) and then stepwise regression or C_p selection is used to choose possibly influential subsets from this base set. If a dummy variable is retained then its corresponding row merits special attention.

These techniques have a certain appeal because they consider more than one row at a time and because of the computational procedures that are available to perform stepwise and all possible subsets regression (Furnival and Wilson, 1974). It is probably best to first choose a set of explanatory variables and stay with them while the dummy variables are selected. Stepwise regression will fail to consider all possible subsets of the dummy variables and may therefore miss some interesting subsets of the base set. All possible subsets regression puts severe limits on the number of dummy variates that can be considered (the size of B*) but branch and bound algorithms can be improved somewhat when only dummy variables of this special form are used. More work needs to be done in this area.

In the special case when just one dummy variable

is added, the t statistic for this new variable is just

$$r_i^* = \frac{r_i}{s(i)\sqrt{1-h_i}} \tag{15}$$

which we have called a studentized residual (Hoaglin and Welsch, 1978). As we can see from (1) or (5) looking at r_i^* (or just r_i) is not sufficient for diagnosing influential observations because a large change can occur when $|r_i^*|$ is small and h_i is large (h_i is always ≤ 1) or conversely. This implies that the general technique, with more than one dummy variate, may not be adequate either and so we turn to some other approaches.

3.2 Derivatives

We now try to generalize (1) and (5) to the larger subset situation. To motivate this we consider infinitesimal perturbation via derivatives rather than deletion. In particular, we will alter the weight attached to the i^{th} observation by replacing $\text{var}(\varepsilon_i) = \sigma^2$ with $\text{var}(\varepsilon_i) = \sigma^2/w_i$. Differentiation of the regression coefficients with respect to w_i, evaluated at $w_i = 1$, for $i=1,\ldots,n$, provides a means for examining the sensitivity of regression coefficients and fitted values to a slight change in the weights given to each observation. The i^{th} component of this derivative is

$$\frac{\partial b(w)}{\partial w_i}\Big|_{w=1} = (X^TX)^{-1}x_i^Tr_i \tag{16}$$

which we combine into the $p\times n$ matrix

$$C \equiv (X^TX)^{-1}X^TE$$

where E is a diagonal matrix with entries $r_1, r_2, \ldots r_n$.

For the fitted values we obtain

$$\frac{\partial Xb(w)}{\partial w_i}\Big|_{w=1} = X(X^TX)^{-1}x_i^Tr_i \tag{17}$$

which can be combined into the $n\times n$ matrix HE.

Our concern is with subsets of observations that have a large influence. One way to identify such subsets is to consider the directional derivatives, $(Cv)^T$, or for the fit, v^TEH, where v is a column vector of unit length with nonzero entries only in the rows of a subset, D, of the base set which we want to perturb. Since we are interested only in the extreme case, we are led to consider

$$\sup_v v^TC^TCv \quad \text{and} \quad \sup_v v^TEHEv \tag{18}$$

where we have used the fact that $H^2=H$. These suprima are given by the largest eigenvalues of the matrices $[C^TC]_D$ and $[EHE]_D$ where the subscript is used to denote the matrix formed by considering only the rows and columns in D. These computations will, of course, have to be done for all of the subsets in the base set up to some given size (perhaps half the size of the base set).

In the above discussion we could have replaced the partial derivatives (16) by the difference approximation, b-b(i). For the fitted values this leads to a natural generalization of the statistic developed by Cook (1977).

Computing largest eigenvalues is, in general, relatively expensive. However, there are several ways to reduce the cost. For convenience we will only consider the matrix corresponding to fitted values, HE. Let F be the submatrix formed by considering only the columns of HE which are in a subset D of B*. F can be decomposed into two matrices Q and R as was discussed in section 2. (These are not the same matrices, however.) Given R, Cline et al. (1977) have devised a way to approximate the largest singular value in $O(p^2)$ operations. R is updated (or downdated) as D is changed by one row in $O(p^2)$ operations. Thus the search over subsets of size $1,2,\ldots,k^*$ from the base set of size m is of order

$$p^2\sum_{k=1}^{k^*}\binom{m}{k} \leq p^2 2^m \tag{19}$$

which is still fairly large. We are seeking a better method for this problem.

A lower cost approach makes use of the idea that the trace of the square of a matrix reflects the magnitude of the larger eigenvalues. This trace is simply the sum of squares of the elements of the matrix, $(EAE)_D$. Thus for each subset D we need to compute

$$T(D) = \sum_{i,j\in D} h_{ij}^2 r_i^2 r_j^2 = \sum_{i\in D} h_i^2 r_i^4 + 2\sum_{i<j} h_{ij}^2 r_i^2 r_j^2. \tag{20}$$

Given T(D) for a subset D of size d, T(D') for D' the same size as D but with one row changed can be obtained in O(d) additions. Similarly, if D" has one new row added to D, then T(D") is obtained in O(d+1) additions. The total operation count for subsets of size $1,2,\ldots,k^*$ is of the order

$$\sum_{k=1}^{k^*} k\binom{m}{k} \leq k^*2^m \tag{21}$$

and these are essentially all additions.

In the single row situation, the largest eigenvalue for the fit matrix is $h_i r_i^2$ when derivatives are used and $h_i r_i^2/(1-h_i)^2$ for differences.

It is also useful to consider putting the same weight on all rows in D. That is, we will perturb all of the rows in D simultaneously and by the same infinitesimal amount. For the fit derivatives (17) the chain rule implies that we need to compute the quadratic form

$$r_D^T H_D r_D$$

where $r = (r_1,\ldots,r_n)^T$. This is of the same order as (21) and is once again all additions.

3.3 Covariance

It is easy to construct examples where the deletion of a subset of observations does not affect b very much, but has a large effect on the variance of b. These points are worth knowing about, because they represent situations which may be used to increase the precision in future experiments. It is also important to know how the variance is affected for subsets that do have a disproportionate influence on the estimated parameters or fitted values.

One way to measure the change in variance is to compare the two matrices $s^2(X^TX)^{-1}$ and $s^2(D)(X^T(D)X(D))^{-1}$ in the determinantal ratio

$$\text{COVRATIO} \equiv \left[\frac{\det[s^2_{(D)}(X^T(D)X(D))^{-1}]}{\det s^2(X^TX)^{-1}}\right]^{1/p} . \qquad (22)$$

Computation of this ratio is facilitated by the fact that

$$\frac{\det X^T(D)X(D)}{\det X^TX} = \det (I-H)_D \qquad (23)$$

where $(I-H)_D$ again stands for the submatrix formed by considering only the rows and columns of I-H that are contained in D. In addition if we let Z = [y:X] then

$$\text{COVRATIO} = \frac{(n-p)}{(n-p-d)} \frac{\alpha(D)}{[\det(I-H)_D]^{(p+1)/p}} \qquad (24)$$

where

$$\alpha(D) = \frac{\det Z^T(D)Z(D)}{\det Z^TZ} = \det(I-P)_D \qquad (25)$$

and $P = Z(Z^TZ)^{-1}Z^T$. Andrews and Pregibon (1977) have developed algorithms for computing quantities like $\det(I-P)_D$ based on the Choleski decomposition of I-P which can compute all subsets of k^* in order

$$\sum_{k=1}^{k^*} k^2\binom{m}{k} \le (k^*)^2 2^m .$$

COVRATIO can therefore be obtained at about twice that cost.

When looking only at single rows,

$$\text{COVRATIO} = \frac{1}{\left(\frac{n-p-1}{n-p} + \frac{r_i^{*2}}{n-p}\right)(1-h_i)^{1/p}} \qquad (26)$$

which is based on r_i^* and h_i. This points out the utility of looking at r_i^* and h_i separately and in combinations like (26).

3.4 Geometry

The principle reason for looking at multiple-row procedures is to deal with masking. However, we may also want to see if an influential subset can be divided into subgroups of similar nature. This is particularly interesting when the observations cannot be grouped on the basis of prior knowledge (i.e., time) or when there is prior knowledge but unexpected groupings occur.

In this section the focus is more on geometric outliers rather than on influential observations because we will use the matrix Z formed by adjoining y to X. Thus y loses its special place except where significance levels for some of these statistics are computed.

Consider two groups of observations D (of size d) and G (of size n-d). Let v_1 be an n×1 vector consisting of ones for rows contained in D with zeros elsewhere and v_2 be the same for G. The Wilks' Λ statistic for comparing D and G is (Rao, 1965, p.484)

$$\Lambda(D) = \frac{\det\left[\tilde{Z}^T\tilde{Z} - \frac{1}{d}\tilde{Z}^Tv_1v_1^T\tilde{Z} - \frac{1}{n-d}\tilde{Z}^Tv_2v_2^T\tilde{Z}\right]}{\det(\tilde{Z}^T\tilde{Z})} . \qquad (27)$$

For computation, several simplifications are possible. First, note that $(v_1+v_2)^T\tilde{Z} = 0$ since the data has been centered. Thus the numerator of (27) is equivalent to

$$\tilde{Z}^T\tilde{Z} - \frac{n}{d(n-d)}\tilde{Z}^Tv_1v_1^T\tilde{Z} . \qquad (28)$$

We now use the fact that when c_1 and c_2 are column vectors

$$\det(I-c_1c_2^T) = 1-c_2^Tc_1 . \qquad (29)$$

From (28) it is clear that

$$\Lambda(D) = \det(I - \frac{n}{d(n-d)}(\tilde{Z}^T\tilde{Z})^{-1}\tilde{Z}^Tv_1v_1^T\tilde{Z}) \qquad (30)$$

and then (29) may be used to show

$$\Lambda(D) = 1 - \frac{n}{d(n-d)}v_1^T\tilde{P}v_1 \qquad (31)$$

where

$$\tilde{P} = \tilde{Z}(\tilde{Z}^T\tilde{Z})^{-1}\tilde{Z}^T . \qquad (32)$$

Fortunately $v_1^T\tilde{P}v_1$ is very easy to compute since it involves sums of elements of the projection matrix $\tilde{P}$. We generally examine the smaller values of $\Lambda(D)$ (corresponding to the fact that D is not like the rest of the data, G) for subsets D of B^* of size $1,2,\ldots,k^*$. The computational details are similar to those presented earlier for the trace of the square of a matrix and are of the same order (21). Further gains might be made by considering branch and bound methods.

If we assume (only for guidance) that the columns of $\tilde{Z}$ are from a p-variate Gaussian distribution, we can use the fact that

$$\frac{(n-p-1)}{p}\frac{1-\Lambda(D)}{\Lambda(D)} \sim F_{p,n-p-1} . \qquad (33)$$

It is important that not just the smallest value of $\Lambda(D)$ be examined for each group size, k, since there may be several significant groups of this size. Gaps in the values of $\Lambda(D)$ for a given k are also usually worth noting.

When D consists of just the i^{th} row

$$\Lambda(\tilde{x}_i) = \frac{n}{n-1}(1-h_i) \quad \text{and}$$

$$\Lambda(\tilde{z}_i) = \frac{n}{n-1}(1-h_i)/\left(1+\frac{r_i^{*2}}{n-p-1}\right) .$$

The latter is, again, a particular combination of h_i and r_i^*.

References

Andrews, D.F. and Pregibon, D. (1977), "Finding the Outliers that Matter," Technical Report #1, University of Toronto, Graduate Department of Statistics, March.

Cline, A., Moler, C., Stewart, G. and Wilkinson, J.H. (1977), "On an Estimate for the Condition Number of a Matrix," Informal Manuscript.

Cook, R.D. (1977), "Detection of Influential Observations in Linear Regression," Technometrics, 19, pp. 15-18.

Furnival, G.M. and Wilson, R.W. Jr. (1974), "Regression by Leaps and Bounds," Technometrics, 16, pp. 495-512.

Golub, G.H. (1965), "Numerical Methods for Solving Linear Least Squares Problems," Numerische Mathematik, 7, pp. 206-216.

Golub, G., Klema, V. and Stewart, G.W. (1976), "Rank Degeneracy and Least-Squares Problems," Computer Science Technical Report Series, #TR-456, University of Maryland.

Hoaglin, D.C. and Welsch, R.E. (1978), "The Hat Matrix in Regression and ANOVA," The American Statistician, February.

Mickey, M.R., Dunn, O.J. and Clark, V. (1967), "Note on the Use of Stepwise Regression in Detecting Outliers," Computer and Biomedical Research, 1, pp. 105-111.

Rao, C.R. (1975), Linear Statistical Inference and Its Applications, New York: Wiley.

Welsch, R.E. and Kuh, E. (1977), "Linear Regression Diagnostics," Working Paper #923-77, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Wood, F.S. (1973), "The Use of Individual Effects and Residuals in Fitting Equations to Data," Technometrics, 15, 4, pp. 677-694.

Acknowledgements

The authors would like to thank David Belsley, David Gay, Gene Golub, David Hoaglin, Jon Kettenring, Virginia Klema, and Edwin Kuh for helpful discussions and Joan Kargel for the typing. This research was supported in part by National Science Foundation Grant SOC76-14311 to the NBER Computer Research Center and the M.I.T. Center for Computational Research in Economics and Management Science.

NUMERICAL DETERMINATION OF THE DISTRIBUTIONS OF STOPPING VARIABLES ASSOCIATED WITH SEQUENTIAL PROCEDURES FOR DETECTING EPOCHS OF SHIFT IN DISTRIBUTIONS OF DISCRETE RANDOM VARIABLES

by

S. Zacks
Case Western Reserve University
Cleveland, Ohio 44106

ABSTRACT

Algorithms for the determination of the distribution functions of stopping variables are developed, for two types of sequential detection procedures. The first type is a Bayes quickest detection procedure. The second type is based on the crossing of linear boundaries by sample sums. Approximations which are particularly designed for computer application are given for the first type of stopping variables. Wiener process approximations, which significantly reduce the amount of computations required for the other type of stopping variables, are given too.

1. Introduction

In the present paper we discuss the problem of determining the distributions of stopping variables which are associated with two types of sequential procedures for detecting the epoch of shift in the distribution law of a sequence of random variables. One type of detection is based on a Bayesian approach, while the other type is based on a cumulative sums procedure. In the present paper we restrict attention to sequences of discrete random variables having lattice distributions. Without loss of generality, we assume that these distributions are concentrated on the non-negative integers. Let $X_1, X_2, \ldots$ designate such a sequence of observable random variables. It is assumed that the variables $X_1, X_2, \ldots, X_{\theta-1}$ are identically distributed, having a common distribution function $P_0(x)$; the variables $X_\theta, X_{\theta+1}, X_{\theta+2}, \ldots$ having a common distribution $P_1(x)$, and that all these variables are independent given θ. The distribution functions P_0 and P_1 are known, but the epoch of shift θ, is unknown. In the present study we investigate several methods of evaluating the probabilities $P_\theta\{N \geq n\}$, $n = 1,2,\ldots$, where N designates the stopping variables under consideration. These probabilities, which are the tails of the distribution functions of the stopping variables, N, can then be used for the computation of various measures of efficiency, connected with the operating characteristics of the detection procedures. We provide here a short description of the detection procedures under consideration.

We start with the class of Bayes procedures. In the Bayesian framework, the epoch of shift θ is considered a non-negative random variable having a prior distribution $\pi(\theta)$. After the n-th observation ($n=1,2,\ldots$) one computes the posterior probability $\pi(\theta \leq n|\underline{X}_n)$, where $\underline{X}_n = (X_1, \ldots, X_n)$ is the vector of the first n observations. The associated stopping variable N_1 is defined as:

(1.1) N_1 = least integer n, $n \geq 1$, such that $\pi(\theta \leq n|\underline{X}_n) \geq \pi^*$,

π^* is a pre-assigned threshold, which belongs to the interval $(0,1)$, and is usually close to 1. As shown by Shiryaev [2], this type of a sequential myopic procedure is Bayes optimal for certain models.

The other type of detection procedures is based on the cumulative sums $S_n = \sum_{i=1}^{n} X_i$ ($n=1,2,\ldots$). We consider a class of sequential procedures which terminates as soon as the cumulative sums cross some (linear) boundaries $b(n)$; i.e.,

(1.2) N_2 = least integer n, $n \geq 1$, such that $S_n \geq b(n)$.

This is a wide class of procedures, and many procedures can be approximated by stopping variables of the type N_2, with an appropriate choice of boundary $b(n)$. For example, in a detection problem for Poisson random variables, Khan [1] has shown that if the procedure is based on the conditional probabilities $\pi(\theta \leq n|S_n)$ then the boundary for the stopping variable, which terminates sampling as soon as $\pi(\theta \leq n|S_n) \geq \pi^*$, can be very well approximated by a linear function $b(n) = a + bn$.

In Section 2 we develop the Bayesian framework, for the class of stopping variables N_1, more specifically in relation to the type of prior distribution of θ which was employed by Shiryaev [2] in the proof of its optimality. We derive certain recursive formulae which are useful in the

Research supported by the Office of Naval Research under Contract No. NR 00014-75-C-0529, Task NR 042-276, at Case Western Reserve University.

determination of the distribution function of N_1. In Section 3 we develop the distribution function of N_1 and prove that these variables are finite with probability one. In Section 4 we provide a method for the numerical approximation to the distribution of N_1. Section 5 is devoted to the distribution of N_2. In Section 6 we show how the distribution of N_2 can be approximated by that of appropriate stopping times for Wiener processes. This kind of approximation reduces significantly the computing time. This is illustrated in the numerical example of Section 7, which presents some typical results for cases of Poisson distributions.

2. The Bayes Procedures

In the present section we discuss the Bayes procedure and some basic recursive relationships for the determination of the posterior probabilities. Let q_n designate the posterior probability, given the values of the first n observations, that $\theta > n$. That is, $q_n = P[\theta > n | \underline{X}_n]$, $n = 1,2,\ldots$. Let $q^* = 1-\tau^*$. The stopping variable N_1 is defined in terms of q_n in the following manner:

$$N_1 = \text{least integer } n,\ n \geq 1,\ \text{such that } q_n \leq q^* \quad . \tag{2.1}$$

We develop now the basic recursive equation for the determination of the posterior probabilities q_n. The prior distribution of θ considered here is the "unanticipating" prior used by Shiryaev [2]; namely, the geometric prior:

$$\pi(\theta) = \tau(1-\tau)^{\theta-1} \quad , \ \theta = 1,2,\ldots \quad . \tag{2.2}$$

This particular prior distribution has the property that the recursive relation for q_n developed below does not depend on n, but only on the value of q_{n-1}.

Let $p_t(x)$, $x = 0,1,2,\ldots$ be the probability mass function corresponding to $P_t(x)$ $(t=0,1)$, and let $Z_j = p_1(X_j)/p_0(X_j)$, $j = 1,\ldots,n$. According to Bayes theorem the following recursive equation holds:

$$\begin{cases} q_{n+1} = \dfrac{(1-\tau)q_n}{(1-\tau)q_n + (1-(1-\tau)q_n)Z_{n+1}} \quad , \ n = 0,1,2,\ldots \\ q_0 \equiv 1 \end{cases} \tag{2.3}$$

The posterior probabilities q_n can be computed according to a fixed transformation, i.e., $q_n = T(q_{n-1}, Z_n)$ $n = 1,2,\ldots$.

3. The Distribution of N_1 Under θ

Let $H_n(z;\omega)$ be the distribution function (c.d.f.) of $Z_n = p_1(X_n)/p_0(X_n)$, where $\omega = 0$ if $n < \theta$ and $\omega = 1$ if $n \geq \theta$. The c.d.f. $H_n(z;\omega)$ is determined by $P_\omega(x)$. According to (2.3) the posterior probability q_{n+1} depends only on q_n and on Z_{n+1}. Furthermore, given θ, Z_{n+1} is independent of q_n, which is a function of $Z_1,\ldots,Z_n$. Thus,

$$P_\theta[q_{n+1} \leq \tau | q_n] = 1 - H_{n+1}\left(\frac{(1-\tau)q_n}{1-(1-\tau)q_n} \cdot \frac{1-\tau}{\tau} - 0;\omega\right) \quad , \tag{3.1}$$

where in general $F(x-0)$ denotes the left limit of $F(x)$ at x. Introduce the defective distribution functions $K_n(u;\theta)$, where

$$\begin{aligned} K_1(u;\theta) &= P_\theta[q_1 \leq u] \\ K_n(u;\theta) &= P_\theta[q_1 > q^*,\ldots,q_{n-1} > q^*, q_n \leq u] \quad , \ n \geq 2 \quad . \end{aligned} \tag{3.2}$$

According to (2.3),

$$K_1(u;\theta) = 1 - H_1\left(\frac{1-u}{u} \cdot \frac{1-\tau}{\tau} - 0;\omega\right) \tag{3.2}$$

and

$$K_n(u;\theta) = \sum_{q^* < x \leq 1} P_\theta[q_n \leq u | q_{n-1} = x]\, k_{n-1}(x;\theta), \quad n \geq 2$$

where $k_n(x;\theta) = K_n(x;\theta) - K_n(x-0;\theta)$ is the (defective) p.d.f. corresponding to $K_n(x;\theta)$. From (3.1) and (3.2) we obtain for every $n \geq 2$,

$$K_n(u;\theta) = \sum_{q^* < x \leq 1} \left[1 - H_n\left(\frac{1-u}{u} \cdot \frac{(1-\tau)x}{1-(1-\tau)x} - 0;\omega\right)\right] \cdot k_{n-1}(x;\theta) \quad . \tag{3.3}$$

Similarly, the p.d.f. satisfies for each $n \geq 2$ the recursive equation

$$k_n(x;\theta) = \sum_{q^* < y \leq 1} h_n\left(\frac{(1-x)(1-\tau)y}{x[1-y(1-\tau)]};\omega\right) k_{n-1}(y;\theta) \tag{3.4}$$

where $h_n(\cdot;\omega)$ is the p.d.f. of $H_n(\cdot;\omega)$. We express now the probability distribution of N_1, under θ, by the above functions. For every $n \geq 2$,

$$\begin{aligned} P_\theta[N_1 \geq n] &= P_\theta[q_1 > q^*,\ldots,q_{n-1} > q^*] \\ &= K_n(1;\theta) \quad . \end{aligned} \tag{3.5}$$

Furthermore,

$$P_\theta[N_1 = n] = K_n(q^*;\theta) \quad . \tag{3.6}$$

Hence, from (2.3), (3.5) and (3.6)

$$\begin{aligned} K_n(u;\theta) &= P_\theta[N_1 \geq n] \\ &\quad - \sum_{q^* < x \leq 1} H_n\left(\frac{1-u}{u} \cdot \frac{(1-\tau)x}{1-x(1-\tau)} - 0;\omega\right) k_{n-1}(x;\theta) \quad . \end{aligned} \tag{3.7}$$

From (3.6) and (3.7) we derive that

$$\begin{aligned} K_n(q^*;\theta) &= P_\theta[N_1 \geq n] - P_\theta[N_1 \geq n+1] \\ &= P_\theta[N_1 \geq n] - \sum_{q^* < x \leq 1} H_n\left(\frac{1-q^*}{q^*} \cdot \frac{(1-\tau)x}{1-x(1-\tau)} - 0;\omega\right) \cdot k_{n-1}(x;\theta). \end{aligned} \tag{3.8}$$

It follows that

$$P_\theta[N_1 \geq n+1] = \sum_{q^* < x \leq 1} H_n\left(\frac{1-q^*}{q^*} \cdot \frac{(1-\tau)x}{1-x(1-\tau)} - 0;\omega\right) k_{n-1}(x;\theta) \quad . \tag{3.9}$$

Furthermore, since $(1-\tau)x/(1-x(1-\tau))$ is an increasing function of x,

$$P_\theta[N_1 \geq n+1] \leq H^*\left(\frac{1-q^*}{q^*} \cdot \frac{(1-\tau)}{\tau}\right) P_\theta[N_1 \geq n] \quad , \tag{3.10}$$

where $H^*\left(\frac{1-q^*}{q^*} \cdot \frac{1-\tau}{\tau}\right) = \max\left\{H\left(\frac{1-q^*}{q^*} \cdot \frac{1-\tau}{\tau};0\right), H\left(\frac{1-q^*}{q^*} \cdot \frac{1-\tau}{\tau};1\right)\right\}$.

Applying (3.10) recursively we obtain

$$P_\theta[N_1 \geq n+1] \leq \left(H^*\left(\frac{1-q^*}{q^*} \cdot \frac{1-\tau}{\tau}\right)\right)^n \quad , \text{ all } \theta \quad . \tag{3.11}$$

Thus, if $H^*\left(\frac{1-q^*}{q^*} \cdot \frac{1-\tau}{\tau}\right) < 1$ the stopping variable N_1 is finite with probability one, i.e.,

$$\lim_{n\to\infty} P_\theta[N \geq n+1] = 0 \quad , \text{ all } \theta \tag{3.12}$$

and

$$\begin{aligned} E_\theta\{N_1\} &= \sum_{n=1}^{\infty} P_\theta[N_1 \geq n] \\ &\leq 1 + \frac{H^*\left(\frac{1-q^*}{q^*} \cdot \frac{1-\tau}{\tau}\right)}{1 - H^*\left(\frac{1-q^*}{q^*} \cdot \frac{1-\tau}{\tau}\right)} \quad . \end{aligned} \tag{3.13}$$

4. Numerical Approximation To The Distribution of N_1

As seen in (3.9), the determination of the tail probabilities $P\{N_1 \geq n+1\}$, for each $n \geq 2$, requires the determination of the probability functions $k_{n-1}(u;\theta)$ at all the points of increase of $K_{n-1}(u;\theta)$. The difficulty in actual computations is that these points of increase vary from one iteration to another. Moreover, the number of points of increase grows in an exponential manner, and it becomes very tedious and difficult to compute the $k_n(u;\theta)$ functions exactly. We propose therefore an alternative method of computation, which provides a good approximation to these functions and can be performed relatively fast. According to the proposed approximation we partition the interval $(q^*,1]$ into M subintervals of equal length, Δ. We denote by $\xi_i^{(M)} = q^* + i\Delta$ $(i=0,\ldots,M)$, the end points of these subintervals. For $n = 1$ we determine the probabilities

$$(4.1) \qquad G_1^{(M)}(i;\theta) = P_\theta\{\xi_{i-1}^{(M)} < q_1 \leq \xi_i^{(M)}\} = K_1(\xi_i^{(M)};\theta) - K_1(\xi_{i-1}^{(M)};\theta) \quad , \quad i = 1,\ldots,M \ .$$

Then, for every $n \geq 2$ we determine recursively the probabilities,

$$(4.2) \qquad G_n^{(M)}(i;\theta) = \sum_{j=1}^{M} P_\theta\{\xi_{i-1}^{(M)} < q_n \leq \xi_i^{(M)} \mid \tilde{\xi}_j^{(M)}\}\, G_{n-1}^{(M)}(j;\theta) \quad ,$$

$i = 1,\ldots,M$, where $\tilde{\xi}_j^{(M)} = (\xi_{j-1}^{(M)} + \xi_j^{(M)})/2$. The functions $G_n^{(M)}(i;\theta)$ are designed to approximate the probabilities $P_\theta\{\xi_{i-1}^{(M)} < q_n \leq \xi_i^{(M)} \ , \ N_1 \geq n\}$. Accordingly, we approximate the tail probabilities of N_1 by

$$(4.3) \qquad P_\theta^{(M)}\{N_1 \geq n+1\} = \sum_{j=1}^{M} H_n\left(\frac{1-q^*}{q^*} \cdot \frac{(1-\tau)\tilde{\xi}_j^{(M)}}{1-(1-\tau)\tilde{\xi}_j^{(M)}} - 0;w\right) G_{n-1}^{(M)}(j;\theta) \ .$$

In actual computations the values of q_n are determined up to a certain digit after the decimal point, rounding off the remainder. This means that in practice we determine M to be sufficiently large so that the approximation (4.3) becomes an exact determination of the distribution of the stopping time based on the rounded values of q_n. The method proposed here reduces the $\{q_n; n \geq 1\}$ process to a finite state Markov Chain. State 0 is the absorbing state, corresponding to all values of q_n smaller or equal to q^*. The other M states correspond to the subintervals $(\xi_{i-1}^{(M)}, \xi_i^{(M)}]$. This Markov Chain is transient. One could use known results from the theory of Markov Chains to characterize the present approximation to the distribution of N_1.

5. The Distribution of N_2

Since the sample sums $S_n = \sum_{i=1}^{n} X_i$ assume only the non-negative integer values, the recursive equations provided below determine the distributions of N_2 exactly. Define the defective distributions

$$(5.1) \qquad F_\theta(x,n) = P_\theta[S_n \leq x \ , \ N_2 \geq n] \quad , \quad \begin{matrix} x = 0,1,\ldots \\ n = 1,2,\ldots \\ \theta = 0,1,\ldots \end{matrix}$$

and their corresponding probability functions

$$(5.2) \qquad f_\theta(x,n) = F_\theta(x,n) - F_\theta(x-1,n) \quad ,$$

where $F_\theta(-1,n) = 0$. For $n = 1$ we have

$$(5.3) \qquad f_\theta(x,1) = \begin{cases} P_0(x) & , \quad \text{if } \theta > 1 \\ P_1(x) & , \quad \text{if } \theta \leq 1 \ . \end{cases}$$

If N_2 is defined in terms of the boundary function $b(n)$, $n = 1,2,\ldots$, where $b(n)$ are finite positive integer, then by letting $c(n) = b(n) - 1$, one obtains by the usual convolution argument that, for each $n \geq 2$,

$$(5.4) \qquad F_\theta(x,n) = \sum_{i=1}^{c(n-1)} P_{w(n)}(x-i)\, f_\theta(i,n-1) \quad ,$$

$w(n) = 0$ if $n < \theta$ and $w(n) = 1$ if $n \geq \theta$, $P_w(x)$, $w = 0,1$, is the distribution function of X. Similarly, the corresponding probability functions satisfy the equation

$$(5.5) \qquad f_\theta(x,n) = \sum_{i=0}^{c(n-1)} P_{w(n)}(x-i)\, f_\theta(i,n-1) \ , \quad n \geq 2 \ .$$

These functions can be determined (numerically) for any type of boundaries. In the next section we will show that in cases of linear boundaries we can obtain good approximations by considering the crossing times of Wiener processes. For these kinds of crossing times we can provide closed formulae for their distributions.

The tail probability $P_\theta\{N_2 \geq n+1\}$, $n = 1,2,\ldots$ are given in terms of the $F_\theta(x,n)$ functions by

$$(5.6) \qquad P_\theta\{N_2 \geq n+1\} = F_\theta(c(n),n) \ .$$

6. Weiner Process Approximations for Linear Boundaries

In the present section we develop a method for approximating the distribution of N_2 in cases of linear boundaries by a function based on the distribution of a stopping time, τ, at which a Wiener process $\{W(t) \ ; \ t \geq 0\}$ crosses a linear boundary.

Let μ_i and σ_i^2 $(i=0,1)$ denote the expectation and variance of the distribution P_i $(i=0,1)$. We assume that $0 < \sigma_i^2 < \infty$ $(i=0,1)$. Since S_n is a sum of independent increments having distributions P_0 until θ and P_1 after θ, we consider a combined Wiener process $\{W_\theta(t) \ , \ t \geq 0\}$ with $W(0) = 0$; drift

$$(6.1) \qquad \mu_\theta(t) = \begin{cases} \mu_0 & , \quad \text{if } t < \theta \\ \mu_1 & , \quad \text{if } t \geq \theta \end{cases}$$

and variance (per time unit)

$$(6.2) \qquad \sigma_\theta^2(t) = \begin{cases} \sigma_0^2 & , \quad \text{if } t < \theta \\ \sigma_1^2 & , \quad \text{if } t \geq \theta \ . \end{cases}$$

This process is a combination of two Wiener processes $\{W_0(t); \ t \geq 0\}$ and $\{W_1(t); \ t \geq 0\}$ where

$$W(t) = I[t < \theta]\, W_0(t) + I[t \geq \theta]\, (W_0(\theta) + W_1(t-\theta))$$

Consider a linear boundary $b(t) = a + bt$, $0 < a < \infty$, $0 < b < \infty$, and the corresponding stopping time

$$(6.3) \qquad \tau = \inf\{t; W_\theta(t) = b(t)\} \ .$$

According to Theorem 4 of Simons [3], the probability that the Wiener process $W_0(t)$ does not touch the boundary before time t, $0 < t < \theta$, given that $W_0(t) = y$, is

$$(6.4) \qquad P\{W_0(u) < a + bu \ , \ \text{for all } 0 \leq u \leq t \mid W_0(t) = y\} = [1 - \exp\{-\frac{2}{\sigma_0^2 t} a(a + bt - y)\}]^+ \quad ,$$

where $[\cdot]^+ = \max(0,[\cdot])$. Hence, since the marginal distribution of $W_0(t)$ is normal with mean $\mu_0 t$ and variance $\sigma_0^2 t$, if $t < \theta$ we have

$$P_\vartheta\{\tau > t\} =$$

(6.5) $$P\{W_0(u) < a + bu, \text{ for all } 0 \le u \le t\}$$
$$= \Phi\left(\frac{a+t(b-\mu_0)}{\sigma_0\sqrt{t}}\right) - \exp\left(-\frac{2a}{\sigma_0^2}(b-\mu_0)\right)\Phi\left(\frac{t(b-\mu_0)-a}{\sigma_0\sqrt{t}}\right) .$$

$\Phi(\cdot)$ designates the standard normal integral. We notice that it is simpler to determine $P_\vartheta\{\tau > t\}$ for $0 < t < \theta$ according to (6.5), than to determine $P_\vartheta\{N_2 \ge n+1\}$ for $1 \le n < \theta$ according to (3.8). The question is whether $P_\vartheta\{\tau > n+1\}$ yields a good approximation to $P_\vartheta\{N_2 \ge n+1\}$. Since the Wiener process is a continuous time process, if N_2 and τ are determined with respect to the same linear boundary then $P_\vartheta\{\tau \ge n+1\} < P_\vartheta\{N_2 \ge n+1\}$. Thus, we propose the following modification. Determine the tail probabilities $P_\vartheta\{\tau' \ge n+1\}$, $n = 1,2,\ldots,\vartheta-1$, where τ' is the stopping time at which the Wiener process $W_0(t)$ touches for the first time the parallel linear boundary $b'(t) = a' + bt$, $a' > a$. The value of a' is determined so that $P_\vartheta\{\tau' \ge n+1\}$ is equal to $P_\vartheta\{N_2 \ge n_0+1\}$ for some relatively small value n_0. In this way we force the distribution of τ' to be close to that of N_2. Although the determination of a' requires the exact evaluation of $P_\vartheta\{N_2 \ge n_0+1\}$, if n_0 is not large this can be generally performed quite fast.

Thus, if $H(n_0) = P\{N_2 \ge n_0+1\}$, a' is the root of the equation

(6.6) $$\Phi\left(\frac{a'+n_0(b-\mu_0)}{\sigma_0\sqrt{n_0}}\right) - \exp\left\{-\frac{2a'}{\sigma_0^2}(b-\mu_0)\right\} \cdot \Phi\left(\frac{n_0(b-\mu_0)-a'}{\sigma_0\sqrt{n_0}}\right) = H(n_0) .$$

The value of a' can be determined from (6.6) by applying the Newton-Raphson method, or any other convenient method of solution.

We consider now the Wiener process approximation for values of n greater than ϑ. Employing the Markovian nature of the Wiener process, we can write for every $t > \vartheta$, according to (6.4)

(6.7) $$P\{W(u) < a' + bu, \text{ for all } 0 \le u \le t | W(\vartheta) = y, W(t) = z\}$$
$$= [1 - \exp\{-\frac{2a'}{\sigma_0^2\vartheta}(a' + b\vartheta - y)\}]^+ \cdot [1 - \exp\{-\frac{2(a'+b\vartheta-y)}{\sigma_1^2(t-\vartheta)}(a' + bt - y - z)\}]^+ .$$

Since the conditional (marginal) distribution of $W(t)$, given $W(\vartheta) = y$, is normal with mean $y + \mu_1(t-\vartheta)$ and variance $\sigma_1^2(t-\vartheta)$, we obtain for all $y \le a' + b\vartheta$ that

(6.8) $$P\{W(u) < a' + bu, \text{ for all } 0 \le u \le t | W(\vartheta) = y\} =$$
$$\left[1 - \exp\left\{-\frac{2a'}{\sigma_0^2\vartheta}(a'+b\vartheta-y)\right\}\right] \cdot \left[\Phi\left(\frac{a'+t(b-\mu_1)+\vartheta(\mu_1-\mu_0)-y}{\sigma_1\sqrt{(t-\vartheta)}}\right)\right] - \exp\left\{-\frac{2(a'+\vartheta(b-\mu_0)-y)(b-\mu_1)}{\sigma_1^2}\right\}\Phi\left(\frac{t(b-\mu_1)+\vartheta(\mu_1-\mu_0)-2\vartheta b - a'+y}{\sigma_1\sqrt{(t-\vartheta)}}\right) .$$

Finally, the probability of $\{\tau > t\}$, when $t > \vartheta$, is obtained by determining the expectation of (6.8), with respect to the distribution of $W(\vartheta)$. Define the terms

$$\alpha_1 = [a' + \vartheta(\mu_1-\mu_0) + t(b-\mu_1)]/\sigma_1\sqrt{t-\vartheta} ,$$

$$\alpha_2 = [-a' - \vartheta(b-\mu_0) + (t-\vartheta)(b-\mu_1)]/\sigma_1\sqrt{t-\vartheta} ,$$

(6.9) $$k_1 = -\frac{2a'}{\sigma_0^2\vartheta}(a'+b(\vartheta-\mu_0)) ,$$

$$k_2 = 2a'/\sigma_0^2\vartheta ,$$

$$\gamma_1 = \frac{-2(a'+\vartheta(b-\mu_0))(b-\mu_1)}{\sigma_1^2} ,$$

$$\gamma_2 = \frac{2(b-\mu_1)}{\sigma_1^2} .$$

From (6.8) and (6.9) one obtains that, for $t > \vartheta$,

(6.10) $$P_\vartheta\{\tau > t\} = P_1(t;\vartheta) + P_2(t;\vartheta) + P_3(t;\vartheta) + P_4(t;\vartheta) ,$$

where for $B_\vartheta = a' + (b-\mu_0)\vartheta$

$$P_1(t;\vartheta) = \int_{-\infty}^{B_\vartheta} \Phi(\alpha_1 - \beta y)\, n(y|0,\sigma_0^2\vartheta)dy ,$$

$$P_2(t;\vartheta) = \int_{-\infty}^{B_\vartheta} \exp\{\gamma_1 + \gamma_2 y\}\, \Phi(\alpha_2 + \beta y)\, n(y|0,\sigma_0^2\vartheta)dy ,$$

$$P_3(t;\vartheta) = \int_{-\infty}^{B_\vartheta} \exp\{k_1 + k_2 y\}\, \Phi(\alpha_1 - \beta y)\, n(y|0,\sigma_0^2\vartheta)dy ,$$

and

$$P_4(t;\vartheta) = \int_{-\infty}^{B_\vartheta} \exp\{k_1+\gamma_1+(k_2+\gamma_2)y\}\, \Phi(\alpha_2 + \beta y)\, n(y|0,\sigma_0^2\vartheta)dy .$$

The evaluation of the integrals $P_i(t;\vartheta)$ $(i=1,\ldots,4)$ can be carried out in several different ways (infinite series expansions, polynomial approximation to the $\Phi(\cdot)$ function and integration, etc.). When an efficient computer subroutine is available for the calculation of the $\Phi(\cdot)$ functions then, as our experience has shown, numerical integrations provides generally fast and good determination of these functions.

7. Numerical Examples For Poisson Distributions

In the present section we illustrate the various procedures discussed above concentrating on the special case of Poisson distributions. The distribution P_0 is a Poisson with (a known) mean λ, $0 < \lambda < \infty$. The distribution P_1 is Poisson with mean $\lambda + \delta$, where $\delta > 0$. We start with the Bayes procedure.

The likelihood ratio statistic is $Z_n = e^{-\delta}(1+\delta/\lambda)^{X_n}$. Let $F(x;\lambda)$ designate the Poisson c.d.f. at the point x, with mean λ. The c.d.f. of Z_n is

(7.1) $$H_n(z;\omega(n)) = P_\vartheta\{e^{-\delta}(1+\frac{\delta}{\lambda})^{X_n} < z\} = F\left(\frac{\ln z + \delta}{\ln(1+\frac{\delta}{\lambda})}; w(n)\right) ,$$

where

(7.2) $$w(n) = \begin{cases} \lambda & , \text{ if } n < \vartheta \\ \lambda + \delta & , \text{ if } n \ge \vartheta . \end{cases}$$

From (3.2) and (7.1) one can immediately obtain the cumulative distribution, $K_1(u,\lambda)$ of the posterior probability q_1. Similarly, formula (7.1) is used for the evaluation of the conditional probabilities required for the recursive formula (4.2) according to the expression

(7.3) $$P\{\xi_{i-1}^{(N)} < q_n \le \xi_i^{(N)} | q_{n-1} = \tilde{\xi}_j^{(N)}\} = H_n\left(\frac{(1-\epsilon)\tilde{\xi}_j^{(N)}}{1-(1-\epsilon)\tilde{\xi}_j^{(N)}} \cdot \frac{1-\xi_{i-1}^{(N)}}{\xi_{i-1}^{(N)}}; 0; w(n)\right) - H_n\left(\frac{(1-\epsilon)\xi_i^{(N)}}{1-(1-\epsilon)\tilde{\xi}_j^{(N)}} \cdot \frac{1-\xi_i^{(N)}}{\xi_i^{(N)}}; 0; w(n)\right) .$$

Thus, if one has an efficient computer subroutine to determine the cumulative Poisson distribution $F(x;\lambda)$, the functions $G_n^{(N)}(i;\vartheta)$ $i = 1,\ldots,M, n \ge 1$, can be rapidly calculated according to (4.1), (4.2), (7.1) and (7.3). Similarly, the tail probabilities given in (4.3) can be immediately determined.

TABLE 1

VALUES OF $P_\theta^{(M)}\{N_1 \geq n\}$ FOR $\theta = a$, $\lambda = 12$, $\delta = 5$, $\pi^* = .3$ and $q^* = 1.3$ IN THE BAYES PROCEDURE

	M		
n	20	40	60
2	.97872	.97872	.97872
3	.89602	.89927	.90369
4	.79823	.80542	.80969
5	.70246	.71298	.71925
6	.61592	.62899	.63644
7	.53927	.55424	.56264
8	.47196	.48820	.49718
9	.21866	.23119	.23816
10	.07725	.08350	.08714
11	.02529	.02809	.02971
12	.00813	.00929	.00994
13	.00261	.00306	.00331
14	.00083	.00100	.00110
15	.00027	.00033	.00037
CT	53.2	202.0	453.1

In Table 1 we present the probabilities $P_\theta^{(M)}\{N_1 \geq n\}$ for $n = 2,\ldots,15$ for $M = 20, 40, 60$. These values of M were chosen to illustrate the effect of increasing M on the tail probabilities. These probabilities were determined according to the following parameters; $\lambda = 12$, $\delta = 5$, $\pi = .3$ and $q^* = .3$. The shift epoch is at $\theta = 9$. We label this as Case A. We see in Table 1 that in Case A partition of the interval $(q^*,1]$ into $M = 20$ subintervals does not yield sufficient accuracy. The difference between the values corresponding to $M = 40$ or $M = 60$ is considerably smaller. It seems that $M = 60$ provides probability values which are already stable at the second or third decimal place. However, we have to pay for it by a considerable increase of computing time. The values presented in the various tables of this paper were computed on a timesharing computer (Honeywell GE-400). The computing time, CT (in seconds) is given in the last row of Table 1. All computations are in double precision.

In Table 2 the tail probabilities corresponding to $P_\theta^{(M)}\{N_1 \geq n\}$, $P_\theta\{N_2 \geq n\}$ are presented. These probabilities were computed according to the parameter values $\lambda = 12$, $\delta = 8$, $\pi = 3$, $q^* = .7$, $M = 20$, $a = 8$, $b = 12$ and $\theta = 9$. This case is labeled as Case B.

TABLE 2

THE TAIL PROBABILITIES ASSOCIATED WITH THE 2 STOPPING VARIABLES, CASE B

n	$P_\theta^{(M)}\{N_1 \geq n\}$	$P_\theta\{N_2 \geq n\}$
2	.9626	.9884
3	.9135	.9492
4	.8631	.9015
5	.8140	.8552
6	.7675	.8131
7	.7236	.7756
8	.6822	.7423
9	.2274	.3062
10	.0540	.2483
11	.0118	.0954

In Table 2 we obtain some information on the sensitivity of the two detection procedures. We see that the Bayes procedure tends to stop too soon.

We conclude the present section with a comparison of the probabilities $P_\theta\{N_2 \geq n\}$ and the associated Wiener process approximation. We consider only the approximation given by (6.5) and (6.6). In the present case, Case C, we have again $\lambda = 10$, $a = 8$, $b = 12$, and assume that there is no shift in λ. We adjust the approximation so that at $n_0 = 6$ formula (6.5) yields $H(6) = .6253$. Solution of (6.6) according to the Newton-Raphson method (with one iteration only), yields $a' = 16.37$.. The approximation is given in Table 3.

TABLE 3

EXACT VALUES OF $P_\theta\{N_2 \geq n\}$ AND WIENER PROCESS APPROXIMATION $P_\theta\{\tau' \geq n\}$ IN CASE C

n	$P\{N_2 \geq n\}$	$P\{\tau' \geq n\}$
2	.9984	.9953
3	.9659	.9559
4	.8772	.8665
5	.7545	.7492
6	.6253	.6266
7	.5063	.5126
8	.4040	.4134
9	.3200	.3302
10	.2523	.2621
11	.1918	.2072
12	.1566	.1633
CT	285.8	1.0

The probabilities $P_\theta\{\tau' \geq n\}$ were computed according to (6.5) for all $2 \leq n \leq 12$. We see that the approximation is quite satisfactory, especially since the reduction in the computing time is so drastic. After $H(6)$ was determined it took only one second to compute a' and $P_\theta\{\tau' \geq n\}$ for all $n = 2,\ldots,12$. If we add to this the amount of time required to compute first the values of $P_\theta\{N_2 \geq n\}$ for $n = 2,\ldots,6$ (all the $G_n^{(M)}(j;\theta)$ functions have to computed) the total computing time for the approximation should be roughly one half of the total computing time of the exact method. If we obtain a satisfactory approximation when n_0 is smaller ($n_0 = 3$ say) then the saving in computations will be more significant.

8. References

[1] Khan, A. M., On Detecting Changes In The Mean Of A Poisson Process, Ph.D. Dissertation, The Department of Mathematics and Statistics, Case Western Reserve University, 1972.

[2] Shiryaev, A. N., On Optimum Methods in Quickest Detection Problem, Theory of Probability and Applications, VIII, (1963), 22-46.

[3] Simons, G. A., A Class of Sequential Procedures for Choosing One of K Hypotheses Concerning the Unknown Drift Parameter of the Wiener Process, Annals of Mathematical Statistics, 38 (1967), 1376-83.

WORKSHOP 8

HIGH DIMENSIONAL FILES: LARGE OR COMPLEX

Chair: Frank W. Stitt, ALZA Research

SIR, A WORKING SYSTEM FOR MANAGING LARGE AND COMPLEX RESEARCH DATA

Gary D. Anderson
McMaster University

Eli Cohen
Wally Gazdzik
Barry Robinson
Vogelbach Computer Center

Abstract:
The Scientific Information Retrieval (SIR) is a research data base management system designed for use with case oriented research studies with multiple types of records collected on each case. SIR provides the researcher with capabilities for detailed naming and labelling of his data items, for specifying editing criteria to use in detecting incoming data errors and for conducting batch or interactive updates to or retrievals from the data base. The SIR language is based upon that of the popular SPSS system and allows data subsets retrieved from the SIR data base to be written directly to either SPSS or BMDP save files for statistical analysis by those systems. SIR includes a report generator and several descriptive statistical options built directly into the system.

This paper provides examples of the use of SIR to address the following seven areas important to research data analysis: 1) complex data structures 2) description and definition of the data items 3) multiple types of data items 4) insuring data integrity and updating the data base 5) providing adequate data security 6) conducting data retrievals 7) providing interactive as well as batch access to the research data base.

I. Introduction

Many modern research studies involve the need to manage and analyze large and complex data sets. These data sets are characterized by multiple types of data collected on each study subject and by the fact that the data set is dynamic in that it continues to grow over time. In addition, the researcher wishes to perform various analysis of the data as the study progresses. Examples of such data sets are provided in the health sciences, for example, by multi-center clinical trials, patient record studies, health surveys, patient monitoring studies, etc.

The following issues are felt by the authors to be among the major ones which must be dealt with in managing research data sets of this kind.

1) Complex data structure:

The data set will contain varying amounts of data on each study subject due to missing observations or because of different numbers of observations on each subject.

2) Data description:

The ability to provide each data item with a meaningful name and label is important to avoid confusion when large numbers of data items are involved for each study subject. Further labelling the values of each categorical data item provides significantly improved clarity on tabulations produced during analysis. The ability to produce a well organized listing of the data item labels and information about the structure in the data set is also important. Good detailed documentation on the definition and coding conventions of each data item leads to a minimization of confusion in analysis and makes for good communication among individuals involved in the study.

3) Various data types:

Much research data is categorical in type. That is, only a small number of discrete integers (or categories) represent the possible observations on that data item. In such cases, it is generally wasteful of computer storage if these integer values are stored as real numbers which require an entire computer word of storage. Therefore, the ability to specify integer as well as decimal data is important.

The storage and management of character information (e.g. names, addresses or comments) can be important to a research study as well. Finally, the ability to store and manipulate dates and clock times in a standard consistent manner is important, particularly in ongoing longitudinal studies.

4) Insuring data integrity:

Methods of insuring, to the greatest degree possible, that data which is allowed into the data file is correct must be given high priority by the researcher. Although there are no

automatic methods of guaranteeing that no wrong data values are placed into the data file, a number of things can be done to assure a high level of data integrity. Data can be checked for containment within a specified range (continuous data) and for being one of a specified set of allowable categories (integer data). It can also be checked for consistency with other data within the same data record or with data already in the data file.

5) Data security:

Data security is particularly important when identifying information such as names, addresses, social security numbers, telephone numbers, etc. are kept as part of study data file. Sufficient flexibility on security provisions is necessary to allow different users of a data set access to specific subsets of the data relevant to their needs while denying them access to sensitive data not important to their analysis.

6) Data retrieval capability:

The ability to extract desired subsets of the data set, to carry out extensive computations and to display, report and statistically analyze the retrieved data are required capabilities. Retrievals should be possible both sequentially and by direct access to the data on specified individual sets of study subjects. Since the data collected on each study subject may be complex and large in volume, it should be possible to do extensive subsetting and computation within individual subjects as well as between subjects. Further, simple straightforward means of displaying retrieved information in reports is important as well as the ability to interface the retrieved data values with existing statistical systems.

7) Interactive access:

With the increasing availability of powerful and responsive interactive computer systems, it is important that the researcher have flexibility in accessing his data set interactively as well as in the batch mode. Consequently, a good interactive query system is a vital research data processing tool.

II. Terminology

Before proceeding to discuss the SIR system, several items of terminology will be defined to clarify their usage in the sections which follow.

1) Variable:

The individual items on which measurements are taken and recordings are made in the course of the sutdy, are defined to be the study variables. The following attributes are important to properly describe and define a variable:

a) A short descriptive name, a more lengthy label and specific labels for individual values of the variable if it is categorical in type.

b) An identification of the type of data the variable represents (continuous, discrete, character, date or time) along with size and location of its values on the input medium.

c) A list of specific values that are valid for the variable or a range within which values for the variable must lie.

d) A list of values which are to represent missing data for the variable.

2) Record:

The data record is defined to be a group of one or more individual variables which are considered together as a fixed identifiable unit. Records are usually composed of those variables which can conveniently be measured together at a point in time (e.g. the measurements taken on a patient during each visit to a clinic). Many different types of records may need to be defined for a given situation to contain all of the various kinds of data collected on each subject over the course of the study.

3) Case:

We will define a case to be the collection of all records which contain measurements on an individual study subject. The nature of the research study will determine the way that a case will be defined for that study. For example, in a health study, a case is likely to be the individual families or specific individuals involved in the study. In a geographic study, the case might be cities, counties, states or countries, etc.

4) Data base:

The data base is defined in this document to be the total set of all records presently collected on the cases of a specific research study. The concept of data base further assumes that there are well defined relationships between types of records as well as between individual records of specific types within the cases of the study.

III. SIR - A case oriented hierarchical research data base management system

In this paper, we will describe the Scientific Information Retrieval system (SIR)[1] which has been designed for use with large and complex research data sets. This system is fully operational on CDC6000 computers and presently in use at a number of research installations. No capabilities will be described here which are not actually operational at the time this document was written.

SIR is designed for use with case oriented study data. By case oriented it is meant that observations of various types are made on many study subjects (persons, animals, cities, states,

lakes, etc.). For case oriented data sets, SIR provides extensive capabilities to meet all of the requirements outlined in the previous section and provides many additional features as well.

The SIR language conventions are based upon the popular SPSS[2] language syntax. For the researcher who has used SPSS, the use of SIR is a very natural step. SIR also has the ability to create SPSS and BMDP[3] save files directly from SIR retrievals, thus providing an easy link to all of the statistical capabilities of these popular statistical systems.

SIR allows the researcher to establish a hierarchical relationship among data items which have been first grouped into records. A hierarchy is defined here to be a structured collection of records in which one record is said to "own" many other records in a top-down or tree-like structure.

With SIR, the hierarchy is established by the relationships that naturally exist among the data items contained in the various records (e.g. a county belonging to a specific state) or on the basis of how the data is to be used (e.g. to maintain direct links between specific records important to common analysis). By properly designing the hierarchical relationships among various records in the SIR data base, the efficiency of common retrievals can markedly be improved. The hierarchical relationships can be completely ignored, however, if a different structure needs to be imposed for a specific analysis.

IV. Establishing the hierarchical structure in an SIR data base

1) Case identification:

The initial step in defining a SIR data base is the definition of the case structure. Each case must possess a unique identifier which distinguishes it from any other case in the data base. This case identifier (case id) variable is often a sequential number assigned to each case as it is admitted to the study. It may, however, be made up of the subject's name, birthdate, social security number or any other items which together will yield a unique identification value for each case in the study. Within a case, each record must contain a record identification key which consists of:

a) the case identifier,

b) a number identifying which record type it is a member of,

c) zero, one or more sorting values.

The case id value identifies the case to which a record belongs while the record type identifier specifies which record type within the case a record is a member of. The record sorting values distinguish between multiple records of the same type within a case and establish any desired hierarchical structures among the records within the case.

For example, consider a study of the relationship between aptitude tests and students actual performance in the classroom in language and mathematics. Two types of records are collected on each student (case) in the study. The first record type consists of demographic information on the student and aptitude test scores collected at the start of each school year in grades four through eight. The second record type is made of records collected at the close of each term of the school year on achievement during that term.

The following schematic illustrates the layout of records for student #25 who has just completed grade five:

Student #25 (Case #25)

Record Type 1 Aptitude Records	Record Type 2 Achievement Records	
Grade 4	1st term	records in this case
	2nd term	
	3rd term	
Grade 5	1st term	
	2nd term	
	3rd term	

Record Type 1	Record Type 2	
Grade #	Grade #	sort variables
	Term #	

We see, then, that each case will have 5 or fewer aptitude records (i.e. 1 type 1 record for each grade 4 through 8) and 15 or fewer achievement records (type 2). SIR maintains a hierarchical relationship between grades and associated term achievement records for that grade because these variables are named as sort variables within the respective record types. In this case, we can say that a specific aptitude record for a given grade "owns" the associated achievement records for that grade.

V. Data security within SIR

Before describing the process of defining and using an SIR data base, it is appropriate that a few comments on the data base security mechanisms within SIR be discussed. A minimal level of security is required for all data bases maintained by SIR. This minimum level requires that a prospective user know the name of the data base and be able to specify a single global password before access can be gained to the data base.

If he wishes, the data base administrator can specify any one of up to 31 levels (0 to 30) of read and write access security to data items or entire records in the SIR data base. In order to gain access to a specific data item, the user's security password must have a read or write (depending upon which he is requesting permission to do) security level which is as high or higher

than the security level in effect on that item. By judicious application of security levels to the data and by providing different users with passwords having security levels appropriate to their individual needs, the data base administrator can retain complete control over access to data base items.

VI. SIR schema definition process

The SIR schema is defined to be the complete specification of the data base. The schema definition process provides SIR with a description of the case structure, initial estimates of the size of the data base and all individual record and variable attributes.

1) Case definition:

The first part of the SIR schema definition process supplies general information about the data base and the individual cases. Within the case definition section, the user specifies security passwords, data base name, case identifier, expected number of cases in the data base, etc.

For example, the following might describe the case definition section for the student aptitude data base described earlier:

FILE NAME	EDUCAT
PASSWORD	SCHOOL
READ SECURITY	(30)DATABA(0)EVERYONE
WRITE SECURITY	(30)DATABA(0)EVERYONE
CASE ID	IDNUM
RECTYPE COL	5
N OF CASES	5000
RECS PER CASE	100
MAX REC TYPES	9
COMMON LIST	AGE,SEX,IQ
DOCUMENT	THIS STUDY IS DESIGNED TO PROVIDE DATA ON THE RELATIONSHIP OF APTITUDE TESTS TO ACTUAL ACHIEVEMENT. FOR EACH CASE THERE ARE UP TO 5 RECORDS (GRADES 4-8) OF TYPE 1 AND UP TO 15 TYPE 2 RECORDS

The first two statements specify EDUCAT as the name of the data base and SCHOOL as the global password. These two names must be used in requesting access to this data base for all future work with it. The two security commands assign read and write passwords for future use. The password DATABA will be used as a master password by the data base administrator since it has a security level of 30 (the highest level possible) and allows him access to all records and variables in the data base regardless of the security assigned to them. This initial creation run provides only one other security password EVERYONE. This password will be given to users of the system and provides them with a security level of zero (the lowest level available). The data base administrator may assign other passwords with different levels at a later time as required.

The fifth statement names the variable IDNUM as the case identifier. This variable will have a unique value for each case in the data base. The next statement identifies that the fifth column on each incoming data record will contain the number of the record type to which that data record belongs. The N OF CASES command provides SIR with an 'upper limit' estimate of the total number of cases expected in this data base. The next two statements provide an estimate of the average number of records expected per case and the maximum number of record types to be defined for this data base respectively.

The COMMON LIST command specifies variables to be placed in the common information record which is a single record created by the system for each case which contains the case identification variable and any other variables which the user wishes to have convenient access to at any time while retrieving inforamtion from the data base. The DOCUMENT command is used to provide explanatory information on the data base which will be listed with SIR schema listings when requested.

2) Record definition:

Once the case definition section is complete, a separate record definition section must be specified for each record type in the data base. As we noted in Section IV, the student aptitude data base has two record types; an aptitude record type and an achievement record type. Thus, two record specification sections will be necessary.

The following example provides a possible listing for the aptitude record definition section and the beginning of the achievement record definition for this example SIR data base:

RECORD SCHEMA	1,APTITUDE
SORT RECORDS	GRADE
DOCUMENT	STUDENT APTITUDE AND DEMOGRAPHIC RECORD. THIS RECORD TYPE CONTAINS UP TO 5 RECORDS PER CASE (ONE FOR EACH GRADE).
VARIABLE LIST	IDNUM,NAME,BIRTHDAY,SEX,GRADE, IQ,TESTDATE,LANGUAGE,MATH
INPUT FORMAT	(I4,1X,A25,A8,I1,I2,I3,A8,2I3)
DATE VAR LIST	BIRTHDAY,TESTDATE(MMIDDIYY)
COMPUTE	AGE=(TESTDATE-BIRTHDAY)/365.25
VALID VALUES	SEX(1,2)
VAR RANGES	GRADE(4,8)/AGE(5,15)/IQ(60,200)
MISSING VALUES	IDNUM,TO MATH(BLANK)
VAR SECURITY	IDNUM,NAME(30,30)

```
VAR LABELS      LANGUAGE,STANDARDIZED LANGUAGE
                APTITUDE SCORE/MATH,STANDARD-
                IZED MATH APTITUDE SCORE

REJECT REC IF   (AGE-GRADE LT 3)

RECORD SCHEMA   2,ACHIEVE

SORT RECORDS

    .

    .

    .
```

The first RECORD SCHEMA statement in this example indicates to the SIR system that the record definition for Record Type 1 follows. The record type is also given the name APTITUDE so that it can be referred to in later retrievals either by number or by name. The SORT RECORDS command specifies that the records within this record type will be ordered on the variable GRADE. Thus, the record key for records of type 1 will be made up of the case id number, a 1 from the record type column and the grade the student is in when this record is collected. Again, the DOCUMENT command may be used to provide documentary information, this time with reference to this record type.

The VARIABLE LIST, INPUT FORMAT, COMPUTE, MISSING VALUES and VAR LABELS commands are old friends to many as they are directly compatible with their SPSS counterparts. Several new commands have been added, however, to those available within SPSS. For example, the DATE VAR LIST provides for variables to be read as a date string, but to be stored within the system as the number of days from a reference date. VALID VALUES and VAR RANGES allow the user to specify permissable values for discrete variables and ranges for continuous variables respectively. Further data integrity checks can be supplied in the data definition by using the REJECT REC IF command which gives logical consistencies which must be checked on incoming data records before allowing the record into the data base.

The VAR SECURITY command places read and write security levels on variables. Variables have a zero security level by default and, consequently, can be accessed by any user who is permitted access to the data base. When higher security levels are given to a variable, however, the users security level (determined by the security password he provides at his entrance into SIR) must be as high or higher than that present on the variable before he is allowed access to any of its contents. In addition to the variable security command, a record security command is also available which permits security levels to be assigned to the entire record type.

Following the completion of the definition of the first record type, a second RECORD SCHEMA command begins the definition of the second record type. Record Type 2 is named ACHIEVE and contains the achievement scores the student receives at the end of each school term. As we see by the SORT RECORD command, since this record type contains records for different years in school and then for each term within the years, it will be ordered on grade in school (GRADNO) and then on the individual terms (TERMNO) within grades. Thus, the key for records of type 2 will consist of case id, a 2 from the record type column, a grade number and a term number. Because the grade is a sort variable for both record types 1 and 2, the record type 2 records are hierarchically "owned" by the record type 1 records and direct key linkage is retained between a grade record from record type 1 and the associated term records for that grade from Record Type 2.

VII. SIR schema documentation

Once the schema definition has been completed for an SIR data base, the user can request one of several schema listings. The schema listing options produce 8½" x 11" page size documentation listings suitable for binding. These schema listings are meant to serve as reference documents on the study.

The user can choose one or more of the following listing options:

1) STRUCTURE

   This option provides a listing of the contents of the DOCUMENT sections from the schema definition command set along with information on the case structure for the data base.

2) LABELS

   The LABELS option provides a list of variable names and labels from the data base.

3) REGULAR

   This listing includes the variable names, short labels, ranges, missing values, location on the input data record, etc.

4) DETAILED

   This option provides the most complete listing available which adds the complete variable labels and the value labels to those things included in the REGULAR listing.

By default, the schema listing options include all variables from every record type in the data base. However, the user can request specific record types and restrict the listing further to only specified variables if he wishes.

In addition to the schema listing options available, the LIST STATS option allows the user to request, at any time, a complete set of statistics on the status of the data base.

VIII. SIR data input and updating

1) Data input:

   Once the SIR data definition process is completed for at least one of the record types in the data base, data records belonging to that record type may be entered into the data base. The data

records may follow the schema definition deck directly in the same computer run or they may be input from a separate file or even directly from an interactive terminal if desired.

Several input modes are available to provide extensive control over data input:

a) The READ INPUT DATA command indicates both that new data can be added from the incoming records and that old data already in the data file can be over-written by incoming records. Thus, this option allows both adding new data and updating old data in the same run.

b) The ADD REC option allows only new data to be added and will allow no old data to be overwritten.

c) The REPLACE REC option allows only old data to be overwritten and no new data to enter the data base. This option would only be used if records already in the data base were being re-entered to correct some of the values they contain.

d) The UPDATE REC option allows records already in the data base to be updated by placing only the record identifiers and the values to be changed in their correct columns on the new updating record. All other data on that record is left unchanged in the data base. The following schematic illustrates how the UPDATE REC option words:

Variable list....ID	V1	V2	V3	V4	V5
Before updating in the data base 103	7			35	49
Input update record 103		16			48
After updating in the date base 103	7	16		35	48

e) The MODIFY REC and the MODIFY REC IF commands are designed to be used interactively to modify specific variable values in the data base from a terminal. With this command, the user first identifies the record to be changed and then using COMPUTE and IF statements, modifies the variables desired by name.

2) Handling input errors:

Errors can occur in input data records for a number of reasons. The value for a variable may violate the valid values or the range specified for that variable in the schema definition. The value of one variable may be inconsistent with that at another variable. A record may be attempting to overwrite one already in the data base when SIR is in the ADD mode or the opposite situation may be true when it is in the REPLACE mode. Finally, a record may be in violation of a security level established for the data base.

Normally, any record containing an error is not allowed to enter the data base so as to protect its integrity. Instead, a listing of the record containing the offending variable is reported and as much information about the error which caused the rejection as possible is given. SIR may, however, be instructed to save a copy of any records with errors in a separate file for easier modification and re-entry if desired. As an option, the user may further instruct SIR to accept records with errors in ranges or valid values and to place a missing value in the offending variable location. In all cases, the errors are reported to the user so that he may later correct the erroneous missing values placed in the data base.

3) Deleting records from the data base:

Often, for many reasons, individual records or entire cases may need to be removed from the data base. To facilitate removal of unwanted records, several deletion options are provided:

a) The DELETE REC command deletes a single record from the case for which the record identification key is provided.

b) DELETE REC IF deletes all records in a specified record type which satisfy a logical criterion given. For example, if we wished to delete all records from Record Type 6 for which the variable AGE was not known, the following command could be used:

DELETE REC IF 6(EXISTS(AGE)EQ 0)

c) DELETE CASES deletes one or more cases as specified by a case id list or an interval of case ids.

d) DELETE CASE IF deletes all cases that satisfy a given logical expression.

IX. SIR data retrievals

1) Introduction:

The key to the retrieval power of SIR lies in its ability to conveniently extract desired data subsets from a complex multiple record type data base. The retrieved SIR data base subsets consist of fixed length records tailored for statistical analysis either by built-in procedures within SIR itself or by SPSS, BMDP or other existing statistical systems. The records formed as the result of an SIR retrieval may be processed directly by the SIR descriptive statistical options or the SIR report generator or they may be saved as a new SIR data base. Just as easily, they may be saved as an SPSS or BMDP save file where the variable specification information relevant to the host statistical system is passed on directly to the save file. Finally, records formed by a retrieval may be written as a card image file in a form acceptable as input to any statistical program which the user wishes to use.

2) Creating summary records:

Variables created during an SIR retrieval are called summary variables. A summary variable can be a variable moved directly from the data base or it can be a new variable computed from data base variables or other information as part of the retrieval process. The records produced by the retrieval process are called summary records. It is the summary records that are then available for further analysis.

A retrieval is defined by a series of SIR commands forming a retrieval program. When the retireval program is executed by SIR, each pass through the program creates one fixed length summary record composed of the summary variables defined within the retrieval program.

3) Retrieving a single summary record from each case:

As an example of an SIR retrieval, consider the situation where the user of the student data base defined previously wishes to use SPSS to perform a paired t-test on language and mathematical aptitude scores between grades four and eight. The following SIR and SPSS command statements will perform this analysis:

SIR statements:

```
GET FILE        EDUCAT

PASSWORD        SCHOOL

PROCESS REC     APTITUDE,WITH(4)

COMPUTE         LANG4=LANGUAGE;MATH4=MATH

PROCESS REC     APTITUDE,WITH(8)

COMPUTE         LANG8=LANGUAGE;MATH8=MATH

SPSS SAVE FILE  FILENAME=COMT48

FINISH
```

SPSS statements:

```
GET FILE        COMT48

T-TEST          PAIRS=LANG4 WITH LANG8/
                MATH4 WITH MATH8/

OPTIONS         2

FINISH
```

In this example, the first PROCESS REC statement indicates to SIR that record keys are to be used to retrieve the grade four record from among all aptitude (type 1) records for a student. After this record has been retrieved, two summary variables; LANG4 and MATH4, are to be created from the language and mathematical aptitude scores respectively found on that record. The second PROCESS REC statement tells SIR to retrieve the grade 8 record from the aptitude records for the same student. By the same method as before, two variables LANG8 and MATH8 are created from this record. A summary record is now formed which contains the four summary variable values for LANG4 through MATH8. This summary record is then written to a file called COMT48 which is an SPSS save file completely compatible with the SPSS system. The retireval program will be applied repeatedly to each case in the student data base and will create exactly one summary record like the one described above for each student.

The SPSS statements then tell SPSS to perform the desired paired t-test analysis on the data SIR has written to the SPSS save file COMT48.

As a second example, suppose the researcher wishes to perform multiple linear regression to determine the respective ability of age, sex, IQ and aptitude scores from the aptitude record to explain the variability in average achievement scores received by the student. The following SIR retrieval program and SPSS control card deck will accomplish this analysis:

SIR statements:

```
GET FILE        EDUCAT

PASSWORD        SCHOOL

FOR EACH REC    APTITUDE

SELECT REC IF   (GRADE EQ 5 OR 6 OR 7)

MOVE VAR LIST   AGE,SEX,IQ,LANG,MATH

PROCESS REC     ACHIEVE,WITH(GRADE)

COMPUTE         AVGM=MEANR(ACHMATH);
                AVGL=MEANR(ACHLANG)


SPSS SAVE FILE  FILENAME=EDUREG/
                SUBFILES=GRAD(5)GR5
                (6)GR6 (7)GR7

FINISH
```

SPSS statements:

```
GET FILE        EDUREG

RUN SUBFILES    EACH

REGRESSION      VARIABLES=AGE TO AVGL/
                REGRESSION=AVGM WITH AGE
                TO MATH/
                REGRESSION=AVGL WITH AGE
                TO MATH/

STATISTICS      ALL

FINISH
```

In this example, the FOR EACH REC command tells SIR that it is to create a summary record for each record found in the aptitude record type (type 1) that satisfies the SELECT IF command (i.e. that belong to grades 5, 6 or 7). Each time a grade 5, 6 or 7 record is found, the values on that record for AGE to MATH are moved into the summary record and the PROCESS REC command following is executed. This command tells SIR to go into the

achievement record type (type 2) to the records for the same grade as on the type 1 record just processed (e.g. grade 5) and calculate the mean mathematics and language achievement scores over that entire grade and place them in the summary record being created under the names AVGM and AVGL respectively. The result will be a summary record for each grade 5, 6 and 7 respectively for each student in the data base. These summary records are written to an SPSS save file called EDUREG which is divided into three subfiles, one subfile for each of the three grades being analyzed.

The SPSS control cards listed will then perform separate multiple linear regression analysis on the average achievement scores for mathematics and language within each of the three grades 5, 6 and 7.

4) Retrieval procedures available within SIR:

A number of automatic procedures are available within SIR to process the summary records created by a retrieval in addition to the SPSS save file procedure which we have used in both of the examples just discussed. The following is a complete list of the SIR procedures which will act directly on sets of summary records resulting from an SIR retrieval:

a) CONDESCRIPTIVE

This procedure is similar to the SPSS CONDESCRIPTIVE statistical option. It provides summary statistics (mean, standard deviation, minimum, etc.) on continuous variables.

b) FREQUENCIES

Univariate frequency count distributions and histograms may be generated using this procedure.

c) PLOT

The PLOT procedure allows the user to generate two way scattergram plots of two variables with the option of stratifying the plots on a third variable.

d) SIR SAVE FILE

This procedure causes the summary records to be written as a new SIR data base.

e) WRITE RECORDS

With this procedure the user can write the summary records to a card image file with a specified format so that he can later read them into any program he chooses for further analysis.

f) SPSS SAVE FILE

As described in the examples above, this procedure writes the summary records as an SPSS save file for direct processing by the SPSS system. All variable names, labels, missing value designators, etc., which are in effect within SIR for the summary variables involved will also be written to the SPSS save file created by this procedure.

g) BMDP SAVE FILE

As with the SPSS SAVE FILE procedure, this procedure writes the summary records to a file compatible with BMDP save files for direct analysis by that statistical system.

h) REPORT

The REPORT procedure provides capabilities for generating complex multi-level reports from the summary records resulting from a retrieval. The ability to generate sequential reports exists within the SIR retrieval programs directly. Consequently, the REPORT procedure will only be used when the summary records must be re-ordered before generating the report or when the report is of a complex multi-level nature.

5) An example report:

The report procedure is sufficiently important to warrant presenting an example here to illustrate how it might be used. Suppose the researcher wishes to obtain a list, for each student by grade, of his IQ, mathematics aptitude score and his average mathematics achievement score for that year. Also, at the end of each grade, he wants to list the average values of these scores over all students when they were in that grade. The report might look like the following:

MATH SCORE REPORT PAGE 1

GRADE NUMBER 5

STUDENT'S NAME	IQ	MATH APTIT-UDE	AVE.MATH ACHIEVE-MENT
JONES,JAY	83	63	74.8
MASON,SUE	85	68	70.4
ABBOT,JANE	93	78	72.1
PAM,JON	93	81	79.6
JONES,SEAN	94	65	83.3
.			
.			
.			
POWERS,ED	127	94	97.0
ARNOLD,JIM	135	91	93.8
AVERAGES	112	87	82.5

MATH SCORE REPORT PAGE 12

GRADE NUMBER 6

.
.
.

The following SIR retrieval and report procedure could be used to produce this report:

SIR retrieval commands:

```
GET FILE          EDUCAT

PASSWORD          SCHOOL

SECURITY          DATABA,DATABA

FOR EACH REC      APTITUDE

MOVE VAR LIST     NAME,IQ,GRADE,MATH

PROCESS REC       ACHIEVE,WITH(GRADE)

COMPUTE           AVGMATH=MEANR(ACHMATH)
```

SIR report commands:

```
REPORT            FILENAME=GRLIST/
                  SORT=GRADE,IQ,MATH

LEVEL             1,GRADE

HEADING           'MATH SCORE REPORT',
                  11X,'PAGE',PAGE(I3)//
                  'GRADE NUMBER',
                  GRADE (I3)/
                  'STUDENTS',16T, 'IQ',
                  24T,'MATH',34T, 'AVE.
                  MATH'/
                  'NAME',24T,'APTITUDE',
                  34T, 'ACHIEVEMENT'/
                  '--------------------'
                  '--------------------'

COMPUTE           IQSUM=0;APTSUM=0;
                  ACHSUM=0;N=0

AT END
COMPUTE           IQA=IQSUM/N;APTA=
                  APTSUM/N;ACHA=ACHSUM/N

WRITE             '--------------------'
                  '--------------------'
                  /'AVERAGES', 16T,IQA(I3),
                  24T,APTA(I3),34T,ACHA
                  (F5.1)

LEVEL             2

WRITE             NAME(A15),16T,IQ(I3),
                  24T,MATH(I3),34T,
                  AVGMATH(F5.1)

COMPUTE           IQSUM=IQSUM+IQ;APTSUM=
                  APTSUM+MATH;ACHSUM=
                  ACHSUM+AVGMATH;N=N+1

END REPORT

FINISH
```

No effort will be made in this presentation to explain in detail the statements in this SIR report example. For more information on the REPORT procedure, the user is referred to the SIR Users Manual.[1]

X. SIR utilities

In order to provide the user with the ability to store his SIR data base to tape, to obtain raw listings of the contents of an SIR data base, to produce merged or subseted SIR data bases or to transport an SIR data base to a foreign computer, a number of utility options have been provided. The following is a complete list of utilities which are a part of the SIR system:

1) TAPE STORE

   This utility stores the SIR files for a given data base to magentic tape.

2) PURGE SIR FILE

   Using this utility, the user can purge an entire SIR data base from the computer disc.

3) SIR CARD DUMP

   This utility is designed for transporting the SIR data base between foreign computers. It dumps the data base to a file in a card image format which can then be read directly into another computer.

4) SIR FILE LIST

   Often the user may wish to dump portions of the SIR data base contents directly to an offline printing device. This utility provides this ability without the formality of defining a specific SIR retrieval.

5) SIR SUBSET

   The user can produce a new SIR data base, using this utility, which in one of several ways is a direct subset of the initial SIR data base. For example, he can produce a data base containing only one of the many record types that may be in the master SIR data base. On the other hand, he could produce a subset SIR data base that contains all of the record types within each case, but only contains a sample (say 1%) of the cases from the master file. Subset files are often useful for testing retrievals before running them against the master file.

6) SIR MERGE

   Using this utility, the user can add the contents of one SIR data base to another SIR data base. Very often, for example, it is useful to add the summary records from a retrieval back into the master file as a new record type to facilitate further analysis.

XI. The SIR QUERY subsystem

The query subsystem provides the SIR user with an interactive, terminal oriented subsystem which makes SIR a true blend between an interactive and a batch system. Using the interactive editor facility built into the query subsystem, the SIR user can:

1) Build his own procedures which may then be either executed directly from the terminal at that time or stored within the data base users procedure file.

2) Execute procedures which have been previously stored within the data base user procedure file.

The editor capability within the SIR query subsystem is a fully implemented, line number

oriented editor featuring searching, tab setting, and line modification operations. The editor may be considered as a work area, similar to its use with BASIC language systems, within which programs may be built and from which they may be executed, stored, etc.

Within the query subsystem, SIR retrieval programs may be built and executed which interactively prompt the user for appropriate input. The interactive program may, in turn, call other previously defined procedures stored in the data base and pass parameters to these called procedures as necessary. Thus, a user can build a set of retrieval procedures or reports specific to his data base which can then be executed as needed directly from within the data base itself.

XII. Conclusion

In this paper we have discussed the Scientific Information Retrieval system. SIR has been designed for use by researchers to manage complex case oriented research data sets of either large or small size. Although SIR was designed with the needs of the health sciences researcher specifically in mind, its application extends to all areas of research where multiple types of data are being collected on many cases over time.

It has been impossible to go into detail in this short document on the large number of commands, special functions and operations available within the SIR system. The interested reader is encouraged to request further detailed information directly from the authors.

References

1) SIR, Scientific Information Retrieval; USERS MANUAL, Robinson, Anderson, Cohen and Gazdzik, SIR Inc., Box 1404, Evanston, Ill., 60204, 1977.

2) SPSS, Statistical Package for the Social Sciences, Second Edition, Nie, Hull, Jenkins, Steinbrenner and Bent, McGraw-Hill, 1975.

3) BMDP-77, Biomedical Computer Programs, P-Series, Edited by W.J. Dixon and M.B. Brown, University of California Press, Berkeley, 1977.

4) A Language for Management of Non-Rectangular Case Oriented Data Files for Statistical Analysis, Anderson, Cohen, Gazdzik, Robinson, Proceedings of the Statistical Computing Section of ASA, 1976, pp.58.

DATA MANAGEMENT IN SAS AND INTERFACES TO OTHER SYSTEMS

A. J. Barr
SAS Institute

Abstract

In one system, SAS offers data management, statistical procedures, and a report-writing language. This integrated approach means that the individual SAS user (or the organization using SAS) needs a smaller mass of knowledge for effective data analysis, since language conventions, data element standards, and error conditions are handled consistently across all three application areas. Features of the data management system are discussed and the counterparts in SAS to the Relational Data Model are mentioned.

1. Counterparts in SAS to the Relational Data Model

The SAS data management facilities manipulate rectangular data sets, which are also called "flat files." In the SAS vocabulary, the columns of the data set are called "variables," and the rows are "observations." Codd's Relational Data Model (1) has brought added appreciation to the elementary data organization used in SAS, which is augmented by several operators.

In (3), Date gives four examples of the relational operators' use. Below, Date's examples are coded in SAS to show analogous capabilities. The data sets used in these examples are:

```
DATA SUPPLIER;
   LABEL SNO=SUPPLIER NUMBER
         SNAME=SUPPLIER NAME;

   INPUT SNO $  SNAME $  STATUS  LOCATION$;CARDS;
         S1     SMITH      20    LONDON
         S2     JONES      10    PARIS
         S3     CLARK      20    LONDON

DATA PARTS;
   LABEL PNO=PART NUMBER
         PNAME=PART NAME;

   INPUT PNO $  PNAME $  COLOR $  WEIGHT  LOCATION$;CARDS;
         P1     NUT      RED        12    LONDON
         P2     BOLT     GREEN      17    PARIS
         P3     SCREW    BLUE       17    ROME
         P4     SCREW    RED        14    PARIS

DATA SP;
   INPUT SNO $  PNO $  QUANTITY;CARDS;
         S1     P1        30
         S1     P2        20
         S1     P3        40
         S2     P1        30
         S2     P2        40
         S3     P2        20
         S3     P3        30
```

Example 1. Find the location of the supplier S1. This is the RDM Select operator. In SAS we code:

```
DATA SUBSET; SET SUPPLIER; IF SNO='S1';
```

giving the data set SUBSET,

SNO	SNAME	STATUS	LOCATION
S1	SMITH	20	LONDON

Example 2. Find SNO and STATUS for suppliers in London. This uses the RDM Projection Operator. In SAS we code:

```
DATA; SET SUPPLIER; IF LOCATION='LONDON';
   KEEP SNO STATUS;
PROC SORT;BY SNO STATUS;
DATA LONDON;SET; BY SNO STATUS;
   IF FIRST.STATUS;
```

giving,

SNO	STATUS
S1	20
S3	20

Example 3. Find PNAME for parts supplier by supplier S1. This uses the RDM Join Operator. In SAS the MERGE statement performs the similar operation:

```
DATA TEMP;SET SP;IF SNO='S1';
DATA TEMP2;
   MERGE TEMP PARTS;BY PNO;
PROC SORT;BY PNAME;
DATA PARTNAME;SET TEMP2;BY PNAME;
   IF FIRST.PNAME; KEEP PNAME;
```

```
giving,   PNAME
          NUT
          BOLT
          SCREW
```

Example 4. For each part supplier, find PNO and names of all locations supplying the part. This uses both RDM Join and Projection operators.

```
DATA TEMP; MERGE SP SUPPLIER; BY SNO;
   KEEP PNO LOCATION;
PROC SORT;BY PNO LOCATION;
DATA TEMP;SET TEMP;BY PNO LOCATION;
   IF FIRST.LOCATION;
```

```
giving,   PNO  LOCATION
          P1   LONDON
          P2   LONDON
          P3   LONDON
          P1   PARIS
          P2   PARIS
```

Chamberlain (2) lists positive attributes of the Relational Data Model, and these attributes also apply to the SAS data management scheme:

1 Simplicity. Users see a single, consistent data structure.

2 Data Independence. The user's program is independent of the way the data is stored.

3 Symmetry. Data base systems that are based on connections between records make some questions easier than others. In a hierarchical data base, questions that start by asking conditions of the root and progress out the branches to the leaves are easily answered. "Questions not reflecting this preferred structure can be asked awkwardly if at all. Since information is represented by data values in relations, there is not a preferred format for a question at the user interface."

4 Strong Theoretic Foundation. Chamberlain points out the mathematical foundation of the Relational Data Model. The SAS data management operators are not derived so formally, but are rather concepts that evolved to meet the needs of our users.

These arguments tend to support the rectangular file approach when it is augmented by a good set of data management operators. But there are benefits from hierarchical organizations as well. The above arguments fail to recognize that there is always some organizational structure to complex data that is important to the updating, maintenance and report preparation processes. Also, redundant data must be stored to act as keys for joining the data sets. In a hierarchical data set, the name of the parent need not be stored with the data for each child's record.

2. Editing and Updating

The SAS programming language is used for editing data for errors. For example, to check for an invalid age variable for college students, this statement could be written:

```
IF AGE 16<OR AGE>80 THEN ERROR AGE=;
```

This statement would set an error flag if the AGE variable were not in the acceptable range, and would also print out the value of the AGE variable and print the input record.

SAS provides two ways to correct errors that are detected: a merge updating method and an on-line interactive update procedure.

The merge updating method has traditionally been used by business. For this method, both the old master and the transaction data sets are sorted by a common key. A merge process matches the transaction records to the records of the old master by comparing the sorted key data element. After all the transaction data is applied to the old master record, the new master record is written on the new master data set. It is possible to add and delete observations with this method and to alter the default updating action with SAS programming statements.

For interactive updating, SAS provides the EDITOR procedure, which works under the Time Sharing Option (TSO) of the IBM Operating System. The procedure works in a direct access mode, so that only the data being modified is read or written. The user can display, update, delete, or add data with EDITOR.

3. Hierarchical Data Sets

Because the SAS INPUT statement can be used as a programming statement like the GET statement in PL/I, SAS can read existing data sets which have a hierarchical structure. Examples of this kind of data set are the Public Use Samples of the Bureau of the Census and computer performance data. SAS also handles files containing many different record types easily: an INPUT statement reads enough of the record to determine the record type, then another INPUT statement reads the variables in the format specific to that record type.

To aid in processing large diverse data sets, SAS breaks them down into useful subsets. Thus such a data set is usually stored as several SAS data sets. When data from different data sets must be processed together, the SAS MERGE statement is used. This approach to large data files increases the efficiency with which large volumes of computer performance data can be processed, since many SAS data sets can be constructed in one pass of a raw data set.

4. Computer-Produced Data Set Documentation

SAS data sets are self-defining, with the data set dictionary stored with the data. This means that the data set cannot ever become separated from its dictionary. Besides protecting against dictionary loss, this scheme also protects against the equally bad situation in which the dictionary does not exactly match the data set. The answers one gets in this situation may look normal, making the error hard to detect.

When many people access the same data, documentation describing the data base is essential. The CONTENTS procedure in SAS provides such a description of a SAS data set. This description includes the variable names, variable labels, input and output formats, and length and storage mode of the variables. CONTENTS also prints a history of the data set, including the programs that produced the data set, and the date and times these programs were executed. CONTENTS

also prints a physical description of the SAS data set, including the tape or disk volume of the data set, the data set name, the amount of disk space used, and the cost to store the data set.

The information produced by CONTENTS is the same as that maintained by the "data dictionaries" associated with large data bases. The Osiris CODEBOOK LIST (4) is a very similar features to CONTENTS; it produces the value labels and what ranges of the variables are in error.

5. Access Method

Basic to the SAS approach to data management is the concept of multiple data sets, and SAS maintains collections of data sets. Before the 1976 implementation of SAS, we used the access methods available with the IBM Operating System. However, the limitations of this method were frustrating. For example, only one member of a partitioned data set could be written at a time. When a member of a partitioned data set was deleted, the space occupied by the deleted member was not available for other uses until a separate job was run to compress the data set.

Consequently, for SAS76 we implemented an access method that enables many data sets to be written concurrently. Its disk management is governed by one simple rule, "When more space is needed for a SAS data set, use the first empty track in the physical data set." With this approach, most SAS users never need to do any space maintenance of SAS data sets.

The problem of directory block allocation is also solved because the directory of a physical data set that contains many SAS data sets is itself a SAS data set. This means that any number of SAS data sets can be stored in a physical data set, and no allocation of directory blocks is necessary.

Although SAS data sets are basically sequential, a direct accessing scheme is maintained. Each observation of a SAS data set can be located directly by its observation number, and the SAS statistical procedures make good use of the direct access mode, as does EDITOR. The statistical procedures sometimes also create temporary data sets which are written in the work space along with SAS data sets, and thus do not require a separate allocation.

In addition to the disk access method, there is a similar access method for storing SAS data sets on tape. Because it is usually not feasible to store data sets containing over a million records on disk, they are stored on tape in most cases.

6. Data Elements

SAS has both numeric data elements and character data elements with lengths up to 200. The numeric data elements are stored as floating point numbers with up to 17-digit accuracy. Some storage compression can be achieved by truncating the floating point numbers. A two-digit number can be stored as a floating point number with one byte for the exponent and one byte for the mantissa.

SAS allows 28 different missing values to be maintained. These missing values are coded by using the 255 non-zero exponent combinations along with a zero mantissa; the IBM 360/370 series never generates these combinations as a result of floating point operations. Our programs can test for missing values very efficiently: if the number has a non-zero floating point value, it cannot be missing. If it tests as zero but the exponent is non-zero, the number is a missing value.

Standards for date, time, and date-time data elements have recently been implemented. Time is uniformly stored in units of seconds. Dates are stored as the number of days since January 1, 1960, and date-time is stored as the number of seconds since January 1, 1960. These conventions make it wasy to calculate differences in times. For inputting, printing, and manipulating date, time, and date-time data elements, SAS offers a sizable collection of functions and format routines.

7. System Implementation

SAS compiles the user's data management and programming commands and generates machine instructions, which are then executed directly by the computer.

Some other systems are implemented through an interpretive mode of execution. In these systems, the compiler produces output that acts as input to another program. Glass (5) compares the timing for executive compiler-generated machine code and his interpretive code, and finds from 2.5 to 33 times faster execution of machine code. This efficiency improvement extends the size of the files that may be efficiently maintained and processed sequentially by as much as 20 to 30 times.

8. Report Writing

Report writing in SAS is an integral part of the SAS programming language. The report writer is used to print raw data, as well as statistics derived from the data. If it were not present, some other programming language would be needed: the user would thus need much more knowledge to produce a report based upon data stored in SAS.

In our 1972 system, we always had a hard time trying to get users to save costs by storing their data in SAS. The most common reason users gave for not storing their data in SAS was that they thought they never could get it back out of SAS. The report-writing ability is a straightforward way of copying a SAS data set into a file that most Operating System programs can process.

9. Interfaces with Other Systems

Automated interchange of information among the major statistical systems is possible because of their self-defining file structures. The SAS procedure CONVERT, which was originally developed so that data stored by the 1972 version of SAS could be processed, was later extended to convert data stored by the BMDP, SPSS, AND OSIRIS systems.

The "stereo sound system" modular approach to data analysis is preferred by many researchers. This approach takes the data base system best suited to the user's needs and interfaces it with a report writer and a statistical system. In response to many in the pharmaceutical industry using the INQUIRE data management system, a procedure for converting data stored in INQUIRE into a SAS data set was developed.

Many users want to use BMDP programs on SAS data sets. The SAS procedure BMDP copies a SAS data set into a BMDP save file, and the BMDP program is executed by the SAS supervisor. In this way, many BMDP programs

can be executed in the same SAS job. This procedure was recently improved by eliminating the step that copies the SAS data set into a BMDP save file. The BMDP program is now loaded from the library and the caller patches into the incore version of the BMDP program. When the BMDP program calls its "get data" subroutine, the call is diverted into the SAS code, which fills the BMDP program's data area with the SAS data. This change required that the linkage editor symbol dictionary of the BMDP program be read from the load module in order to find the address of the "get data" subroutine.

References

Codd, E. F. "A Relational Model of Data for Large Shared Data Banks." Communications of the ACM 13 (June 1970), pp. 377-397.

Chamberlain, D. D. "Relational Data-Base Management Systems." ACM Computing Surveys 8,1 (March 1976), pp. 43-66.

Date, C. J. "Relational Data Base Concepts." Datamation, (April 1976), pp. 50-53.

Glass, R. L. "An Elementary Discussion of Compiler/ Interpreter Writing." ACM Computing Surveys 1,1 (March 1969), pp. 55-77.

Institute for Social Research, University of Michigan. "OSIRIS III, Volume 1, System Programming Description," pp. 187-200.

EFFECTS OF LARGER FILES AND MORE VARIED USAGE ON THE DEVELOPMENT OF THE P-STAT SYSTEM

Roald Buhler, Shirrell Buhler
Princeton University Computer Center
Princeton, New Jersey 08540

ABSTRACT

The size of "large" files used in statistical computing has grown steadily over the last 15 years. This paper describes the evolution of P-STAT as file sizes grew from a few boxes of cards to multiple tapes of data.

1. INTRODUCTION

A general statistical computing system attempts to be reasonably usable over a large area of applications. Problems can surface when the area includes extremes such as simple batch versus prompting interactive, or core resident files versus census files. It requires a tremendous amount of work to augment a batch system with features such as prompting, error correction and screen controls which are obligatory for effective interactive use. For example, verbosity levels are relatively unimportant in a batch run, but become vital in interactive sessions.

An equally major series of changes has been caused by the need to process larger and more complex files than were envisioned ten or fifteen years ago. This paper will outline our experience with large files, starting with the IBM 650 and including the current changes going into P-STAT 78.

2. FILE GROWTH IN THE LAST 15 YEARS.

The size of files processed in university computer centers has increased sharply over the last 15 years for the most basic reasons of all, economic and technical. In 1959 the most common university computer was an IBM 650, which usually had 2000 words of storage (on a drum), no tapes or disks and no on-line printer. Most software was in an assembly language called SOAP (although an early Fortran could be used awkwardly). With no mass storage, a study doing a number of analyses on 10,000 cases usually meant five boxes of cards being read at a very slow card reader again and again. In those days a large file was more boxes of cards than you cared to carry.

Princeton shifted from a 650 to an IBM 7090 in 1962 (after some transitional use of a CDC 1604). The 650 to 7090 shift was a factor of 15 in memory size and 200+ in speed. Fortran II became easily usable and there were tapes for mass storage. One could put 30,000 cards on an unblocked tape and files of this size were considered quite practical. Large was more tapes than you cared to keep track of (for example, 2).

Putting one's cards onto tape and running jobs from the tape had one major effect on statistical software: errors could no longer be corrected by hand. As long as one fed boxes of cards into a reader for each job, it seemed natural to hand correct errors and feed the cards in again. Not so with tapes. The data was already out there, let programs do the checking and either fix or skip over any problems. P-STAT, in 1964, already had

case ID checking and card order checking of multiple card per case files. By 1966 we were able to catch invalid numeric fields, keep control, report the error and convert it to missing. Both of these were results of the "large" files of that era. 10,000 cases with 8 cards per case was large.

By 1968, there was sufficient use of 10,000 case files in P-STAT that the I/O to FIND and SAVE a file became excessive. FIND means locate a given file on a P-STAT system save tape and copy it onto a scratch disk or tape by itself for use in a run. SAVE is the reverse, take a new P-STAT file and copy it from a scratch unit to a P-STAT save tape. Assume one FINDs file X, does a NOOP (which reads a file, applies recodes, etc., and makes another file of the result) and SAVEs the result. Each of the 3 steps reads and writes a file. If X had 10,000 rows, then 60,000 reads or writes of rows would occur.

This, like drought and pestilence, tended to hold file sizes down (or cause complaints) so ASSIGN and ATTACH were put into P-STAT. ASSIGN=XX, TAPE=8$ directed P-STAT to put file XX, when it was created, directly onto unit 8 as a save file. ATTACH=X, UNIT=8 $ caused X to be used directly from that save tape. Thus, using ATTACH, ASSIGN and then the NOOP allowed a 30,000 case file to be processed in the same amount of I/O as a FIND, NOOP, SAVE sequence on a 10,000 case file.

The result was inevitable, users praised ASSIGN/ATTACH briefly, increased their notion of what was a large file, and began once again to push at P-STAT's revised limits. Another point, ASSIGN/ATTACH was a change in design philosophy - it meant that users could direct the system a little bit instead of letting the system manage everything in a general but somewhat inefficient manner.

Computing power continued to grow, of course, as did I/O speed, so by 1972 a file of 200,000 cases was large to a general package but not impossible. An indication of the change in attitude was a long distance call one Friday afternoon from a P-STAT user who had read 95,000 of 110,000 cards when he crashed with a B37 (ran out of disk space). We proposed writing and dictating a quick program to finish the 95,000 case P-STAT file cleanly so all would not be lost, but he declined, saying that the load was light on the weekend and he would redo the entire run with more disk space. Clearly his concept of large was greater than ours.

3. PROBLEMS CAUSED BY LARGE FILES

The time and cost of processing large files are obvious problems. Accuracy and formatting must also be considered.

Crosstabulation in P-STAT allows either an unweighted or a weighted cell N. If both are wanted, the normal N is unweighted and the weighted N is captured by asking for the SUMS of the WEIGHT variable. One government project had a file of 200,000 or so cases with integer weights to represent the entire U.S. population. A table was run with Ns and with sums of the weights in the cells. The Ns were fine. The sums of the weights, however, did not quite add up, i.e., the row total of 143,214,732 was not quite the sum of the four eight-digit row elements. The problem, of course, is that 7 digits is the best one can do in 32 bit single precision on a 360. A new option, integer sums, was quickly spliced into P-STAT to utilize the 9 digit accuracy of a 32 bit integer. (It is interesting, people like data to be consistent even though it is most unlikely to be literally correct to 9 digits.)

Numbers of this size also caused formatting problems. Code had to be added to allow optional wider table cells so that large numbers could be printed in a readable manner.

Clearly large files strain general systems somewhat. A patching solution like integer sums to lessen strain is not ideal but not too bad (except for the ICL 1900, which has 48 bit reals and only 24 bit integers).

4. EXPERIENCES WITH COMPLEX FILES

Our approach to complex file structures has been pragmatic. P-STAT allows only rectangular system files but there can be a number of them simultaneously available to various file manipulation commands. SORT, JOIN and COLLATE are typical. Usually either row labels or data values can be used for sorting or link-ups between files.

```
SORT, IN  = XX ,OUT = XX.SA,
      VAR = SEX / AGE $
```

This sorts file XX and calls the result XX.SA, sorting on age within sex.

```
JOIN, LEFT = DAY1, RIGHT = DAY2,
      OUT  = DAY12 $
```

JOIN combines data from DAY1 and DAY2 into a single file named DAY12. DAY1 and DAY2 must have the same number of rows. The row labels of the rows being joined must match in order to be sure that data of the same case are joined together. Other options

allow the checking to be done on variables, or to be bypassed.

```
COLLATE, LEFT = CHILD,      RIGHT = MOTHER,
         OUT=CHILD.MOTHER, RIGHT.MULT.MATCH,
         VAR = REGION / CITY / DISTRICT /
               HOUSE.NUMBER / AGE.MOTHER,
         LEFT.UN.MATCHED = EXTRA.CHILD $
```

COLLATE always has two input files, identified by LEFT and RIGHT because that is the relationship their data will have in the output file. COLLATE is a JOIN of the subset of matching rows.

This illustrates a COLLATE of a file of perhaps 15,000 children called CHILD with a file of 5,000 mothers called MOTHER. The output file (to be named CHILD.MOTHER) will have a row for each child who matches up with a mother. The output variables will be the child's variables followed by its mother's variables.

The assumption is that the input files are in sort order on the five VAR= variables. Both files have those five variables. RIGHT.MULT.MATCH indicates that a given right (i.e., MOTHER) row can be joined to each of several children that match her on the VAR= variables. The unmatched children (i.e., unmatched rows from the left file) will make up another new output file named EXTRA.CHILD.

Two other commands, DUPLICATES and SUB.STATS permit subgroup aggregation to be achieved. COLLATE can then graft group results to the data of each group member.

5. AN EXPERIENCE WITH A LARGE FILE

About a year ago we spend a week in Turkey helping to produce 130 fairly complex crosstabulations for the Syrian census bureau. About three months before leaving we were told that the file had 700,000 cases and 45 variables. On a 370/145 with 256K, two tapes and two disks, that seemed to us a very large file. At perhaps 200,000 cases per P-STAT system tape, it looked like 4 tapes would be needed, a disturbing thought.

The data was mostly small integers, so a months work on packing and unpacking ensued. We developed some code that compresses strings of repeated values (usually chunks of missing data) and/or packs many small integers into a word. It began working fairly well. The size reduction was about 70 percent so that the 700,000 cases might just fit on one tape. Packing entailed a one time cost when making the file, but large files tend to be made once and then used numbers of times. Therefore, the critical cost was unpacking the file to use it. The extra CPU time it took to unpack the file was balanced by less I/O, so that the net cost of unpacking was zero. After all this work we learned that the file would only be 97,000 cases, clearly not a large file, so we put the packing code away for a while.

Another change that did get included in P-STAT was the ability to select and process the raw input in chunks. If one has input records for 700,000 cases, only a very brave person should try to process all of them in one great gulp and make just one P-STAT file. It is better to process the first 100,000 cases and call the result F1, the next 100,000 becomes F2, etc. The resulting files can be concatenated when used (TABLES, IN=F1+F2+...) with no loss of efficiency. The change that was needed for that second file was the ability to locate card 100,001 quickly. The system indicator is....

```
FIRST.RECORD = 100001,
 LAST.RECORD = 200000,
```

The fastest practical way we know of (in portable Fortran) to skip 100,000 records is....

```
    DO 40 J = 1, 5000
40 READ ( JTAPE, 50 )
50 FORMAT ( /// /// /// /// /// // // )
```

The 97,000 cases did have some structure: the head of the house always immediately preceded the rest of that household. The head had a 1 on variable HH.HEAD, other family members had a zero. For some tables it was necessary to carry the head's occupation, which was in variable HEAD.OCCUPATION, down to the other household members' records (who had missing on HEAD.OCCUPATION). P-STAT's temporary data area was used as follows....

```
( IF HH.HEAD .EQ. 1, SETX .T11.
  TO HEAD.OCCUPATION )
( SET HEAD.OCCUPATION TO .T11.)
```

Whenever a household head occurs, his occupation score is moved into temporary location number 11. That score is then moved back (unnecessarily) for the head, but it is also moved into place for the following household members. Normally P-STAT sets all temporary locations to missing before each row is read. That initialization was dropped for the Syrian runs.

We did have an embarassment. One table was occupation code (200 levels) by a school/education variable (90 levels) by sex. Normally all surfaces (and if possible additional tables) would be done in one file pass. However, 200 by 90 was so large that

only one surface would fit in core during a file pass. We decided to include all three possible levels of missing data (TABLES,.....MISSING=ALL) since a 203 by 93 surface was not that much larger.

The data was on tape (using ATTACH) so we watched it take one pass for males and another pass for females. Then it started a third pass. Third pass? Oh yes, a three level table produces a surface for the total population and then a surface for each value in that third level variable, so the first pass was total, the second was male and this is the female pass. Complete comprehension as it finished the third pass, great consternation as it began the fourth, confusion during the fifth and recriminations during the sixth. Finally we realized what was going on, it was the MISSING=ALL effect. TABLES, as instructed, was making one extra pass for each possible type of missing data on the third variable. Fortunately we do not have 26 types of missing data.

Actually, TABLES did a reasonable thing as it finished the 6th pass. It noted that three unproductive passes had occurred and quit, so we were in fact protected from 23 more.

6. CURRENT P-STAT ENHANCEMENTS

This year we find ourselves working at both extremes inproving both large file performance as well as small file interactive capability. For example, READ is a new, extremely simple command for entering free format data. Every imaginable speed versus human factors decision is tilted towards human factors. It would be a slow way to read 100,000 cases, but that is not the intent of the program.

LIST is a new command for printing a file. It is also totally aimed at attractive printout rather than raw speed. These efforts reflect our conviction that (1) more and more statistical computing will be done interactively in the next 5 years, and (2) that attractive, easy to read output is what our customer's managers like best.

We believe that increasingly large files will be routinely processed interactively because more results per pass of the data can be obtained that way. The user is there to decide what to do as the data unfolds instead of submitting guesses to a batch run.

This makes it rewarding to increase the payoff of a pass through the file and vital to increase the speed of a pass, both for cost and for response time.

Cost of a pass.

Minimizing cost comes from areas like packing (mentioned above), hierarchical structures and, of course, writing highly efficient code.

We recently processed a file of 150,000 cases and 6 variables. The rows in P-STAT files begin with a 16 character row label. These are lovely for combining files and identifying cases. We find, however, that they are almost never used in large, thin files. Sixteen characters is 4 words, each variable is a word, thus 40 percent of that file was unnecessary row label text. Clearly we could benefit by having varying length row labels (of which 0 is one optional length). Packing could have combined the 6 integers into one control word and one packed word. These two improvements would yield an 80 percent reduction in size, extremely important if the data is online for interactive use.

We have done less work with hierarchical structures because we are less sure how to do it. There are certainly two reasons to work in this area. Space and speed improvement is one reason, but we feel the bigger gain in that area will come from packing. The more important reason is between-row data modification. The head of household occupation code mentioned above is an example. For the next year or two our strategy, besides work on packing for large and/or sparse files, will be to enhance the current command language to allow modification within hierarchically related groups of rows. The goal is effective use of data. The internal file structure is less important.

Payoff of a Pass.

It is common for a general system to parse a crosstab program into two parts. One part reads the file and collects a coreload of information representing dozens of tables. The second part does the printing. This gets the most out of the file pass itself and no obvious improvements to the file pass suggest themselves. A more interesting issue is how to get the most out of the resulting coreload of information.

The most fundamental difference between programming a batch system and an interactive system is the handling of user errors. When an error occurs in a batch run there is little to do except explain the problem as clearly as possible, give up on the current command and try the next command.

In an interactive run it is better to prompt for a correction and to keep on going. However, it is not quite that simple.

When the command is typed, no checking occurs until it ends. Then, if a syntax error is found, the user can type FIX$. This brings in a prompting text editor. FIX works well. We feel no need to prompt and correct syntax errors as they are typed.

Some commands, when they begin to execute, need records to complete the command definition before any files are read. For example, a discriminant command might contain NG=6, which means that six groups are involved and definition records are needed for them. Is prompting and on the fly correction necessary if a typing error is made while entering these records? Possibly, but it is not vital as long as the command and the already correct command completion records are retained and can be reprocessed easily, bringing the user back to the point of the error. None of the user's effort is lost, and there was no file processing investment.

Suppose, however, a TABLES command is issued, table definitions are entered (T = AGE BY SEX WITHIN EDUCATION) and the program passes the file and produces the table. The interactive user is then given a number of choices. He can quit (leave TABLES and begin another command), convert the table surface into a P-STAT system file, display it again, send it to a different printer, compress and relabel some of the rows or columns, etc. Keeping control when an error occurs is critical here. If the program gives up because a syntax error is made, the file pass may be wasted because the information now in memory is lost. The test version of P-STAT 78 already has increased support for this type of interactive usage.

7. CONCLUSIONS

There are three areas in which large files (.2 to .5 million cases) are now affecting P-STAT. One is better usage controls. For example a user can now specify MISSING=3/3/0 to control third level missing surfaces in TABLES. Another is better pay-off from a file pass, mainly possible in interactive use. The third is a more efficient form of file structure. Packing would raise the practical limit of P-STAT file size to a million or two. Beyond that, the economics are such that a potential user should consider software tuned for his particular file, either a special purpose package or, conceivably, a general system which has been modified for that one particular file.

MANAGEMENT OF COMPLEX DATA STRUCTURES IN A CLINICAL RESEARCH ENVIRONMENT

Michael A. Fox
Departments of Biomathematics,
City of Hope National Medical Center
and
University of California at Los Angeles

ABSTRACT

The interplay of data management and statistical software is discussed with emphasis laid on practical problems of interfacing systems designed to perform separage but complementary functions.

Hierarchical, relational and inverted structures are reviewed and implications of their implementation are discussed within the framework of actual studies. The management of these studies has all been conducted with A Clinical Information System (ACIS) which, as a compiler, generates custom PL/1 programs. Such programs allow the power of context dependent variables to be used in conjunction with mixed data structures to facilitate highly specialized retrieval strategies.

Both the incorporation within ACIS of the Systematized Nomenclature of Medicine (SNOMed) as a medical language and the practical aspects of translating encoded information into English are featured.

Statistical packages have traditionally been designed to accept data for analysis from a rectangular matrix where each case consists of a single row of the matrix. Analysis is performed on numerical information and in most cases categorical information too must be numeric.

Information systems, in contrast, attempt to preserve the user's concept of data which is often more complex than a simple rectangle and in which the variables may be represented linguistically rather than numerically ("male"vs. 1). This dichotomy emphasized the complementary nature of data management and analysis. It is therefore more sensible to explore the interface between these activities than either to ignore the existence of one or to totally embed one within the other.

Admission of data structures of greater complexity than rectangular arrays exposes two important facets. There is, on the one hand, their internal representation in the machine which often has implications on both the mode and efficiency of retrieval, and on the other there is the attempt to preserve the user's view or model of his data and hence his ability to interact with it.

Application of A Clinical Information System (ACIS) has, to date, been exclusively with biomedical data and it is appropriate to use these data to illustrate the system. A case is a patient on whom information from multiple groups of variables is obtained. For each case there is usually a fixed set of demographic variables (race, sex, date of birth...). Other groups of variables are not fixed but may repeat, the multiplicity of a particular group being arbitrary. Some variable groups may be functionally related to others and some may be independent. Thus microorganisms are determined from blood, sputum or urine cultures. Not only may the number and type of cultures differ from patient to patient, but the number and variety of microorganisms found will undoubtedly be non-uniform. Further, procedures such as operations can be considered as functionally independent of the cultures performed. Viewed as a case the data is naturally structured hierarchically, that is, there is a patient from whom samples are taken and for each sample the microorganism composition is found. Additionally operations and observations may be performed on the patient, and these are separate branches in the hierarchy.

The variable groups (i.e., the set of urine cultures) considered across patients are entities on their own and this leads to a different view of the data and hence a different model. The laboratory handling only the culture material would view the data as a two-dimensional rectangle or array, where each row is a separate culture and each column is a variable of interest. The segregation of data into rectangular arrays represents the relational model. Each "relation" is a set of variable groups. Provided certain linking variables are preserved (deletion of a culture could without the deletion of the microorganisms associated with the culture lead to an inaccessible group of variables), relations and sub-relations are consistent and constitute an ordered graph with no isolated nodes. Such models are important theoretical tools and their attractiveness as practical systems stems from the universality of a schema which operates on rectangles to produce other rectangles. But, as will be later demonstrated, a balance has to be drawn between theoretical schemes and natural representations.

Data retrieval is often concerned with the extraction of particular variables from cases that manifest specific attributes. If information is stored either hierarchically or relationally, then the investigation of an association between types of microorganisms and particular biopsy findings, to use one example, would require a forward or direct search of all

the bio material in order to select those cases in which
the mic rganism content should be examined. Determin-
ation o he set of patients who satisfy more complex con-
ditions ll often require multiple searches. These may
become ensive with large data-bases.

The nverted structure is an additional structure
applied the data which allows complex logical operations
to be pe ormed without reference to the actual data. For
each ca ory of interest a membership or index file is
created or example, there may be cases with a particular
patholo al condition (determined by biopsy), with a spe-
cific d nosis recorded, but with no evidence of a spe-
cific m oorganism (determined by culture). These cases
can be nd by operations on the simpler inverted struct-
ures. A orward pass is only necessary to extract the
require ariables on that subset of patients with the
requeste attributes.

One f the strengths of ACIS is its ability to build
mixed o ultiple data structures. The particular struct-
ures obt ned are determined by choices within the system's
Data De ition Language (DDL). As different systems have
their ov preferred terms equivalents will now be defined.
A tuple r row) of a two-dimensional array is a relation
in a re ional data-base and is referred to as a variable-
group ir CIS, where the members of the tuple are the var-
iables. domain is the set of values that a particular
column c a relation can assume. These values are the cate-
gorical lues in an inverted structure and are collectively
termed a inversion in ACIS, (i.e., "male" and "female" are
the cate ries of the inversion Sex). A sub-relation which
is a rel ion in its own right but contains a linking domain
leading om another relation is a contained variable group
in ACIS.

By e time a potential user of ACIS has collected or
is about o collect data he has a concept of how his data
is organ ed. Thus a block of data on a form (or even the
whole fo) is a variable group and one block may or may not
be hiera nically related to another block. Certain variables
are cate rical and may be useful for segregating or retri.
ing data they become the inversion variables. This concept-
ual view s refined by the user when he describes it to ACIS
using th Data Definition Language. It should be empnasized
that the ata-base can be built in stages and not all variable
groups n d be completely defined initially.

Our g the compilation phase ACIS operates on the user
supplied lauses and generates a viable PL/1 program which
incorpor es the variable-group names and variable names chosen
by the u r. The program produced is link-edited to the nucleus
system a is ready for use.

Sin readable PL/1 code is produced, the user may add
simple l es of code to those sections which operate on the
data to corporate features not supported by the Data Definition
Language For example, the values of the context-dependent
variable can be evaluated: elaboration of this will occur later.

The ata Definition Language is composed of commands which
operate tween variable groups (giving the group-by-group
relation dependency), commands which apply to the data base
as a whc , and commands which supply information with respect
to the p ticular variables within a variable group. Thus, in the
example

```
TIENT:  CONTAINS (BIOPSY, CULTURE...):
        CONSISTS OF (...,SEX(16,1,I),...CHART(3,6,N3),...
        KEY (CHART)::
```

PATIENT the name of the variable-group. CONTAINS is the
command ich builds the directive links to the variable
groups (sub-relations) Biopsy, Culture. CONSISTS OF is
the comm d which specifies the variables (tuples) which
constitu the variable-group (relation). Following each
variable e a series of parenthesized parameters which
provide formation both for the external form of the var-
iable an for its internal packing (e.g., date goes from
the conv ctional mm dd yy to a two byte Julian represen-
tation). Other parameters are used when the variable is to
e inver d, to allow certain choices with respect to blanks
nd for ner similar purposes. For instance, SEX(16,1,I)
informs a compiler that the value of the variable called
SEX is t be found in column 16 and is to be treated as a
simple i erted variable. The command KEY identifies the
variable HART as the prime access key to the patient, and
that all first level hierarchies are to have this as an implied domain in hierarchical data-bases and as an actual domain in relational data-bases. Similarly the command MATCH identifies the variable that is used to link variable groups in a hierarchy deeper than the first level. It is an implied domain in a hierarchical representation and an actual domain in both a relation and its immediate sub-relations. If one elects to use the relational structure, the data-base can be operated without reference to implied hierarchies. Thus repeated cultures taken from the same patient will appear as individual lines in a table of cultures with one particular variable (CHART) being the same. Analyses of these tableaux is thus possible without reference to the rest of the data-base.

Since generation of new relations from existing ones by the "cut-and -paste" functions of projection and join are thus understood, then it will be simple to pass rectangular information to statistical packages, except for the conversion of alpha to numeric. It must be pointed out that to date application of ACIS has been with studies that have natural hierarchical structures. With social science data this may be different.

Since ACIS can operate on multiple files, it is possible physically to detach the data from the inverted reference information and to work in principle during retrieval with only references to cases and with no actual data on the cases being within the machine. This would be important for huge files where post-processing of the reference information would determine which disc packs need be mounted for retrievals requiring actual values of variables.

The building of a data-base consists of adding new cases, adding to established cases, and modifying existing cases. During an ACIS run an audit is maintained of the data and rejected data is reproduced together with error messages. The retrieval program is created using the same Data Definition Language as that used to build the data-base.

No single program can fully encompass the diverse requirements of different studies. In an ACIS medical record file containing some 30,000 cases periodic retrievals are required which take the form of tabulated reports. Although the methodology of retrieval is uniform for the system, particular programs were constructed to provide customized features around a core program. These sub-programs are run as batch jobs. Individual retrievals are conducted interactively. The current system solicits commands from the user. Such commands perform global functions which give information about the data-base as a whole, control the print options, or select either the variables for viewing or the commands which perform the selection and sub-selection operations. Thus if the user accepts the invitation to review the inverted files, an alphabetized listing of the categories comprising the inverted variable chosen will be displayed together with the number of references associated with each category.

An interesting feature of the system is the incorporation of SNOMed, the Systematized Nomenclature of Medicine, as a substructure. This nomenclature, which consists of over 40,000 terms, is divided into six fields: topography, etiology, morphology, function, procedures and diagnoses. Medical English can in most cases be completely encoded into SNOMed and therefore accessed and manipulated by the resulting codes. Automatic translation from information thus encoded is a feature of ACIS. Furthermore, access to all SNOMed terms is available, even when no reference to the particular code exists in the data-base. The following extract is obtained on browsing through a file for the inverted variable F-field or function.

```
F0007     2                    Two people are referenced
          WEIGHT LOSS                        with F0007
F0706     4
          CHILDHOOD
F11553    1
          URIC ACID, NOS*
F1182     3
          GLUCOSE, NOS
F4640     2
          464-467 ATOPIC AND HYPERSENSITIVITY STATES
          HYPERSENSITIVITY REACTION, NOS
          ALLERGIC REACTION (CODE TO E-.....)
```

* NOS means "not otherwise specified"

Having browsed through the variables of interest, either to gain familiarity with the fields or to construct decisions on what to extract, the user can elect to retrieve individual cases, particular sets of inverted variables, or those cases which satisfy a complex Boolean expression whose arguments are categories of inverted variables. Whichever mode of retrieval is being used, the result in the hierarchical schema is cases (or patients). The investigator usually wishes to examine particular variables and these are displayed on the screen, line printer, or data set, dependent on the commands of the investigator.

The following example shows how one patient is retrieved using a Boolean expression; the information retrieved is essentially non-numeric. Analysis of cases similar to this require care. For example, if categorization is performed with respect to microorganism content, then this case could erroneously be classified into more than one class. If, however, microorganisms were scored with respect to seriousness, then such errors would be avoided.

PROFILE	589171	9/22/72	36 M C	
CULTURE	UTI	URINE		
ORGANISM	ASTRGD		**	*This 36 year old male caucasian who died in 1972 was retrieved as one of a set of people with severe hepatocellular changes and severe micronodular cirrhosis.*
ORGANISM	KLEBSI			
CULTURE	UTI	URINE		
ORGANISM	ASTRGD			
ORGANISM	ECOLI			
ORGANISM	KLEBSI			
CULTURE	PERITONITIS	ASCITES/FLU		*(continued) The microorganisms found upon culture, together with the culture site, are displayed for some of his many infections.*
ORGANISM	GRMNRD			
ORGANISM	ASTRND			
ORGANISM	KLEBSI			
CULTURE	PNEUMONIA	SPUTUM		
ORGANISM	PDSARG			
CULTURE	UTI	URINE		
ORGANISM	ECOLI			
CULTURE	PNEUMONIA	SPUTUM		
ORGANISM	GPOSCC			
ORGANISM	GRMNRD			
ORGANISM	GRMNRD			
ORGANISM	NEISSP			
ORGANISM	ASTNGD			
CULTURE	UTI	URINE		
ORGANISM	KLEBSI			

Non-numeric information such as this is extremely interesting to epidemiologists and to those who explore data for its content rather than perform analysis with a specific hypothesis.

The observant statistician will immediately recognize the central problem of interfacing such data to a statistical package. As has been stated before, a case consists of varing number of different repeating groups and with mixed alpha and numeric variables.

The understanding of the implications of both repeating groups and unrelated groups will enable the investigator to construct sensible rectangles by only asking for those variables in which analysis will have meaning within the context of the generated case. It is clear that with a complex data structure the onus rests with the investigator to select variables for interfacing with some thougnt.

The problem of re-coding information from meaningful character strings into numeric categories generally involves only a small number of variables (e.g., "yes", "no", "male", "female"). Such conversions are accomplished using an internal table in conjunction with a variable descriptor vector. Files for the PSTAT System are created by ACIS with this kind of data conversion.

Automatic report generation consists essentially of producing titles, tabulations and totals. ACIS contains a deponent function, the first action of which is to solicit a variable group name and a variable name from the user and then to construct an access path to this variable. Cases retrieved by means already mentioned are then operated upon by a function applied to this additional variable. The command BY induces a sort with respect to the value of this variable, and the sorted cases are then tabulated together with a count of cases with a common value for the sort variable. The deponent function is also used for accepting cases when a particular variable is greater than, less than, or equal to a solicited value.

Practical solutions to interfacing are as follows. Output can be directed to a data set with each variable group preceded by its name. The user has total control therefore over this output with respect to its subsequent disposition, but is left with the task of constructing his own interface. A semi-customized approach where the user supplies a sub-routine is often used. In a study of a few hundred cases of non-Hodgkins lymphomas, the statistical analysis was performed on the survivorship of the patients. Although much information was stored with respect to treatments, classification of lymphomas, and so forth, this was only used to select the cases and was passed to the statistical program as descriptors or classifying variables. The variables used for survivorship were either numeric, or for "alive", "dead", etc. converted to numeric. A routine was constructed to generate the control language for the BMDP statistical program, and for this particular study survivorship was obtained from a two-step job. Thus for a particular study the problem of interfacing may be straight forward, but specialized.

For a more generalized approach there are two options; both of these depend upon rectangularizing the output. The first method relies on a direct interface to BMDQ5S, a generalized rectangularizing program which functionally collapses repetitive values by taking their means, maximum values, etc., and produces a single row consisting of the concatenation of collapsed variable groups composed of the selected variables. The cases so produced are then written out as a BMD Save File. The second method is to expand or pad a case on the left with repetitions of variable groups in their entirety each time a group or level in the hierarchy is retrieved.

In a collaborative study of some 7,000 tissues taken from cases suspected of having Hodgkins disease or non-Hodgkins lymphomas the agreements and disagreements between three separate diagnoses were required. Contributing pathologists sent their tissues and diagnoses to one of twelve regional centers where a second diagnosis was made, after which a diagnosis for each tissue was made at a central repository. Of the many retrievals required the following is a typical example: those cases in which the contributor classified the tissue as depicting Hodgkins disease where there is disagreement between the contributor and both the regional center and the central repository. It will be noted that the agreements and disagreements constitute context-dependent information. Instead of having to retrieve separately on each histology sub-category, status vectors were constructed that automatically reflected the type and nature of the differences. Rapid and effective retrieval was obtained by inverting on these variables. These cases were then automatically sorted into histological subgroups and year of diagnosis using the BY function. The following is part of one line from such a report and illustrates how the pathologist can comprehend his data using abbreviated English as code and representing the agreements with numeric scores.

PATHOLOGY PDD MC MC 76 3 3 5

Contributor diagnosis non-Hodgkins lymphoma poorly differentiated diffuse.

Year

Both center and central repository classify as Hodgkins disease, mixed cellularity.

Status scores showing agreement between center and repository but disagreement with contributor both with regard to main diagnosis (Hodgkins disease and non-Hodgkins lymphoma) and histologic subclassifications. The 5 is a score relative to patterns.

In this study the changes in proportion of agreements over time was of interest. This serves as a measure of the relationship between effective collaboration of experienced contributors with highly specialized training in pathology and the learning process. Real differences expose areas of insufficient understanding of the cellular processes, which will lead to more research to understand the natural history of this disease. A further use of this study is a comparison of the treatment outcomes of those who were correctly classified compared to those who were incorrectly classified. This comparison provides an effective tool for quality control assurance in the diagnosis of this disease.

Acknowledgements

Frank W. Stitt, M.D. of Clinical Sciences, ALZA Research (Palo Alto, California), provided both useful advice on the implementation of the SNOMed interface, and editorial assistance in the preparation of this paper.

The Data Interchange File: Progress Toward Design and Implementation[1]

Richard C. Roistacher
Center for Advanced Computation
University of Illinois
Urbana, IL 61801

Abstract

Most self described data files are designed for maximum efficiency in processing with a particular data management or analysis system. This paper outlines a design for a Data Interchange File for the transfer and archiving of machine readable data. The primary design criteria of the Interchange file are generality, simplicity, and extendibility. The file will accept rectangular and hierarchical files, matrices and tables of arbitrary dimensionality, as well as data from network and relational data bases. The file is self described and will accept machine readable documentation as archival data. Strategies are also outlined for the technical and organizational implementation of the Interchange file.

INTRODUCTION

An increasing amount of scientific research activity involves the dissemination and secondary analysis of machine readable data files. Some of these data files are produced by individual research projects, some, like the Uniform Crime Reports are produced by organizations in the course of their operations; some, like the National Crime Panel Victimization Data, are produced as part of a special research project; while others, such as the National Election Studies, are produced by ongoing data collection efforts funded by a consortium of data users.

A data collector who is also the data's end user has many options for file construction and documentation. In the limiting case, a producer can maintain data in a completely undocumented deck of punched cards, relying solely on memory or a FORTRAN format statement for information about the file. Standardized documentation is not always crucial where data are transferred through personal contact between producers and users, although it is not uncommon to discover a colleague who would be happy to share a file, but who has forgotten its format.

An increasing number of files, however, are transferred not by personal contact between producers, but through dissemination by a central archive. National archives such as the Inter-University Consortium for Political and Social Research and the Roper Public Opinion Center receive data from their original collectors, transform files to a standard form, write appropriate descriptions of files, and fill user orders for data and documentation. Even though it is sometimes possible to refer a client's question to the original producer, no archive can afford to omit information transmitted by the producer from its own file; nor can an archive afford to produce documentation which is anything less than a complete summary of the original producer's documentation.

Card image files. The archivist's problem has in some respects been simplified, and in other respects complicated by the development of integrated statistical systems, self-described files, and machine readable documentation. The universal coin of machine readable data exchange is the deck of punched cards or the unlabeled, unblocked card image tape. The most common representation of data in such files is as numeric characters, with missing data indicated either by blanks or by some arbitrary code, such as a field of nines. In some cases, alphabetic characters are used to indicate valid data values, with "-"s and

---

[1] This work was originally supported by Grant 75-NI-99-0077 from the National Institute of Law Enforcement and Criminal Justice, and is currently supported by Grant 77-SS-99-6021 from the National Criminal Justice Information and Statistics Service, Law Enforcement Assistance Administration.

"&"s used to indicate missing data. A few punched card files contain data coded using tabulating machine methods, in which a data item is always represented in a single column. Arbitrary combinations of multiple punches are used when the standard character set has been exhausted.

Several archives, e.g., the Roper Public Opinion Center and the California State Data Program, maintain their data in card image files after converting alphabetic and multiple punched data items to numeric character form. Almost without exception, data archives will continue to produce card image files for export, even where their internal files are maintained in other formats.

Documentation for card image files is most often in the form of printed entries, giving the name, deck and column numbers, missing data values, and where appropriate, category labels for each variable. Some archives have produced machine readable documentation by punching their codebooks onto cards, which can then be stored and reproduced with the data files to which they refer. Such documentation is both easier to reproduce and more difficult to lose than is paper documentation.

Self-described files. Originally, data files were analyzed with individually written programs designed to no particular standard. Beginning with the Biomedical Data Analysis Program library, (Dixon, et al., 1967), libraries of computer programs with similar control languages and input formats were written at many universities and research centers. In most of these program libraries, and indeed, for many currently used programs, the input data are described with a FORTRAN format statement which is included by the user with the program setup. Such programs and libraries expose the user to the inconvenience and possible error inherent in transcribing codebook information each time a program is used.

Most modern statistical and data management systems use self-described files, which contain program readable documentation. The user of such a system refers to variables by name or number rather than by location in the input record. The analysis program retrieves codebook information from the program readable file description stored with the data. Such systems locate data, provide appropriate handling of missing values, and label both printed and machine readable output with much less user intervention than would be required if a self-described file were not employed.

Even though self-described files make life easier for the user of a particular data management system, they complicate matters for the data archivist, or for the person who wishes to transmit data to someone who uses a different data management system. Most self-described files have been designed to maximize processing efficiency in their "home" systems. In many cases, data are stored in a non-printing internal form, with a high degree of machine and program dependence. Missing data are sometimes represented in program dependent forms which do not fit into the computer's standard set of numeric or character representations. Such files can be called "esoteric," not because they are necessarily incomprehensible, but because they are designed to be read from within a particular system rather than being generally readable. Files which can be interpreted by a simple character dump and which can be read using a FORTRAN-type format statement will be called "exoteric."

A common way of transferring esoteric files is to process them with programs which transform the data into card images and the dictionary into a printed codebook. However, the production of such card-image transfer files undoes much of the work and nullifies much of the value of building the self-described file in the first place. The recipient of a card image transfer file is either reduced to writing FORTRAN format statements, building a new self-described file from the printed documentation, or using a program which attempts to reconstruct a new self-described file from the printed output. The SPSS WRITE FILEINFO subprogram is an attempt to make the process of degrading and transferring an esoteric file as painless as possible. The SPSS procedure, however, is designed to facilitate the transfer of esoteric SPSS files between SPSS installations on different computers, rather than to make the full self-described file available to other data analysis systems.

Other data analysis systems using esoteric files are SAS, the Statistical Analysis System developed at North Carolina State (Barr and Goodnight, et al., 1975), and IMPRESS (Meyers, et al.,1969). A somewhat less esoteric file structure is used by OSIRIS, (University of Michigan, 1976), which stores an esoteric dictionary separately from its data, which are stored as an exoteric file of fixed or variable length character records.

SUPPORT FOR A DATA INTERCHANGE FILE

An increasing number of data management and analysis systems generate and use self-described data files. The development of new systems should be encouraged, for it fosters a healthy diversity and spirit of innovation. Several considerations render impractical any attempt to standardize a common self-described file for all data analysis systems. Several statistical systems which use esoteric files have been in use for many years, and have been used to produce thousands of self described files. It would be impractical to require the users of such systems to learn and adopt a new file format solely for the sake of standardization with other systems. In addition, the use of a standard file for internal processing might require extensive rewriting of existing systems, with the risk of degrading of their internal processing efficiency. Finally, it would be foolish to attempt to restrict the designers of future statistical systems to the limitations of today's data processing techniques.

A better solution is to design an exoteric file capable of supporting most of

the features found in all statistical systems, and designed specifically for the exchange rather than for the processing of machine readable data. Such a file should be designed for simplicity, generality, and extendibility, rather than for data processing efficiency. Designers of statistical systems can accommodate such an Interchange file by writing procedures which convert their own esoteric files to and from data Interchange files. Thus, a data analysis system's own files can be designed to maximize processing efficiency within the system and, can be transformed to Interchange format for transfer and archival purposes.

The CONDUIT conference. In 1974, CONDUIT, the educational computing consortium, held a conference for the purpose of designing a data Interchange file. The conference laid the technical and political ground work for such a file, but lacked the funding for its further development and implementation.

The LEAA Research Support Center. In 1975 the University of Illinois' Center for Advanced Computation, under a grant from the National Institute of Law Enforcement and Criminal Justice, funded a research and development on techniques for archiving and using machine readable social data. One of the tasks of the new Research Support Center was to define an archiving format for data of interest to criminal justice researchers. The best way to accomplish this task was to continue work on the technical and institutional development of a standard data Interchange file. Accordingly, in January 1976, a conference on the exchange of machine readable data was held at Itasca, Illinois. The conference was attended by representatives of the Center for Advanced Computation; the National Institute for Law Enforcement and Criminal Justice, LEAA; the National Criminal Justice and Statistical Service, LEAA; SPSS, Inc.; the Inter-University Consortium for Political and Social Research, developers of the OSIRIS III data analysis system; the Survey Research Center, University of California, Berkeley; The Institute of Statistics at North Carolina State University, developers of SAS; the National Archives; tne Bureau of the Census; and DUALabs.

This is the second in a series of reports resulting from the Itasca Data Interchange Conference. The Interchange file is intended to be used as a standard format for data archives and for the exchange of data between users, regardless of the computing hardware available to them or the data analysis systems they wish to use.

DESIGN CONSIDERATIONS

Several major considerations govern the design and implementation of the data Interchange file. The first consideration is that the file is designed to maximize its utility for data archiving and transmittal rather than for data processing. The file is designed to support arbitrary data structures and missing data representations. It is designed to support more extensive variable and category labeling than is found in most data analysis systems. However, the data Interchange file is not designed to support all features of all existing statistical systems; in particular its features are not necessarily the union of all features to be found in the systems produced by the participating organizations.

The interests of simplicity dictate that the Interchange File support only the minimal number of program readable features. However, it can carry arbitrary textual information in its documentation records. Documentation records can be marked to indicate that they contain information which is not part of the interchange standard, but which is readable to a particular data management system. For example, the documentation records of an Interchange file produced with IMPRESS may carry information on the "standard dichotomy" of each variable. Such information may be read as text by users of systems which do not use such a standard recode, but may be read back into a receiving IMPRESS system. The Interchange standard includes a method for marking documentation records to indicate the presence of program readable information. However, the format of such information is the concern of those responsible for the design of the particular data management system.

Several characteristics are basic to the design and implementation of the Interchange data file.

Character format. Interchange files will be transmitted entirely in character form. It remains to be decided whether both ASCII and EBCDIC will both be allowed as character formats. If there is to be a single character set then obviously it will have to be the American standard ASCII. However, if the capability of almost all machines to understand IBM is granted, then either character set can be allowed. Both the dictionary and the data file will be composed entirely of printing ASCII (or EBCDIC) characters.

Separate dictionary and data files. Each Interchange data set will be transmitted as two separate files, a dictionary and a data file. Thus, it will be possible to separate the dictionary from the data without the use of special programming facilities, and to read and operate on the data either with or without the mediation of the dictionary. One of the major reasons why most self-described files are esoteric is that dictionary and data are written into a single file. Special programming, intrinsic to a particular data management or analysis system, is required to read and interpret the dictionary, and to determine where the dictionary ends and the data begins. Only by storing dictionary and data in two separate files can the need for special programming be eliminated.

Card image dictionaries. Most users of machine readable data have facilities for creating new data variables, facilities which may range from a ten line FORTRAN program to an extensive recode language. However, it cannot be assumed that all users of machine readable data will have access to similarly powerful facilities for modifying and editing

dictionaries. Therefore, it seems wisest to maintain the Interchange dictionary in the form of eighty-column card image character records with five-digit sequence numbers in columns 76-80. Although it is hoped that more elegant facilities will be available, in the last resort a user should be able to produce and edit Interchange dictionaries with tools no more complicated than a set of card listing and punching routines and a key punch.

Free format dictionary records. All dictionary records will have a type identifier in column one, a variable number in columns 2-6, a file identification in columns 73-75, and a sequence number in columns 76-80. In all but the basic variable descriptor record, columns 7-72 will be used to record dictionary information in free format. The use of a free format for recording missing data information, variable, and category labels allows both greater ease of file creation, and greater flexibility in adapting to the characteristics of new statistical systems as they appear on the scene. Free format is obviously easier for someone who must create a dictionary by hand, and is irrelevant to a dictionary creating program. Free format dictionary information is probably easier reading for the person who must interpret the dictionary manually.

It is sometimes argued that the reading and processing of free format information requires unduly sophisticated software and inordinate extra expense. Such criticisms are simply no longer valid. Any reasonable computer system has available the software to do simple parsing of text. The additional expense entailed by the one-time use of parsing programs in converting an Interchange file is negligible in comparison to other expenses incurred in obtaining and processing the file. The length of a dictionary is determined by the width of its data file i.e., the number of variables and their range of codes. Extremely large and expensive files usually contain large numbers of cases as well as large numbers of variables. Thus, dictionary processing expenses for files with large numbers of cases are relatively small in relation to data processing expenses; dictionary processing expenses for one-time conversion from Interchange to some other self-described file format will be negligible in comparison with other expenses incurred in acquiring and processing the data.

Machine readable file documentation. A direct consequence of recording information in free format wherever possible is that extensive file documentation can be produced and transmitted in machine readable form. Each Interchange dictionary will contain a set of documentation records giving technical information on the file. Such information should include the file's name and creation date, the file's logical structure, the system on which the file was originally created, the character set and collating order used in the file, global treatment of blanks, and other technical processing information. The description should also include an abstract of the file and the study which produced it, as well as any notes the producer wishes to pass on to future users.

There are two main reasons why such documentation need not be program readable. First, it is impossible to tell what information producers of Interchange files will want to include in their file description records. Second, most of the information contained in the header records will probably require human intervention in any case. Thus, the interests of flexibility dictate that basic information on such things as a file's name and creation date be acted on by the user, rather than interpreted by a computing system.

Since the contents of documentation records will be transparent to the programs which read and write Interchange dictionaries, there is no reason why they cannot be used to document individual variables and parts of the dictionary. The inclusion of a variable number field on documentation records, in addition to the sequence number, allows unique placement in the dictionary file. Since the documentation text will be in whatever format the producer wishes, all that can be guaranteed is that conversion programs will print all documentation records. However, this does not preclude users from including program readable information beyond that required by the Interchange format.

Variable length data records. Interchange data records will be of variable length in order to support files with more than one type of record. A number of questions regarding data record formats remain to be answered. Although the canonical structure of Interchange data sets is hierarchical, many, if not most, Interchange data sets will be rectangular. Should rectangular data sets be allowed to use fixed length records, or should all Interchange files, regardless of their structure, use variable length records?

Another unresolved question concerns the labeling of Interchange data sets stored on tape. If an IBM tape file is assumed, then it would be expected that IBM labeled, format VB files would be acceptable. If, however, the ANSI tape format is to be used, should the canonical Interchange data format be an ANSI labeled, fixed-length record, blocked file for the dictionary; and an ANSI labeled, variable-length record, blocked file for the data? The problem of users' computers not being able to handle such labeling or deblocking can be solved by writing the labeling and deblocking routines into the programs which import and export Interchange files.

Structure definition. The Interchange dictionary will carry not only a description of each separate type of data record, but also a program readable description of the file's logical structure. A user should be able to obtain a rectangular file based on the lowest level of analysis; e.g., a file in which data from the record on a household has been duplicated and appended to the record of each person in the household.

Standards of Good Practice

Since the Interchange file is designed to maximize flexibility, it will support many options which are not presently in use, and some options which probably should never be used. The final specification of the Interchange data file should implicitly support some rules of good practice, rules whose violation can be supported by the Interchange data structure, but which should be discouraged. While a full list of such rules is probably infinitely long, some rules come immediately to mind.

Although the Interchange data set will store alphabetic data, variables should be stored in numeric form wherever possible. In particular, missing data information should be numeric rather than alphabetic. In some cases it may be necessary to have a variable whose value represents some attribute of a particular value of another variable. For example, variable one may have the value "1" in observations in which the value of variable two is an estimate, while variable one has the value "0" in observations where the true value of variable two has been obtained. Such cases are rare and should be made as rare as possible, since they invariably require some recoding before the file is usable. In most cases identical information can be transmitted using ordinary missing data conventions.

Creators of Interchange data sets should also be sparing in the number of exact-match missing data codes used in the file. Most users receiving a file having, for instance, eight exact-match missing data codes per variable will be forced to spend considerable time collapsing such codes into a more manageable number. File producers should also be discouraged from using mid-range missing data codes. A variable ranging from 0 to 9 should be given a missing data code outside this range, even if a value within the range is unused.

AN IMPLEMENTATION OF THE INTERCHANGE FILE

The Dictionary

All dictionary records are 80 character records containing a type identifier in column 1, a variable number in columns 2-6, a file identifier in columns 73-75, and a sequence number in columns 76-80. The format of columns 8-72 is different for each type of dictionary record.

The sort order for dictionary records should by variable number and then by sequence number. Such a sort order will allow intervals to be left in the original sequence numbers for the insertion of new records.

Documentation records. Documentation records contain free format text giving information about the file as a whole, about sections of the file, and about individual variables and responses. Documentation records dealing with the file as a whole should probably have a variable number of 00000. Since the contents of a documentation record is transparent to the Interchange format, the variable number of a documentation record can indicate the number of the variable to which the record applies, or can be used as an indicator of the record's position in the file.

It seems reasonable that some users would make certain documentation records readable by some receiving programs. For example, a data analysis system which produced value labels longer than twenty characters long might write shortened labels for the Interchange value label records, while inserting the original longer labels in documentation records. The original labels can be marked so that they are automatically recovered if the Interchange file were converted back to its original type. These documentation records would in no sense be a replacement for the Interchange value label records, but would serve as additional documentation for most users, and as a way of allowing some users automatically to recover the original labeling.

Column 7 of the documentation record is used to indicate the presence of text which is program readable. A blank in column 7 indicates that the record carries no program readable information. A printing character in column indicates that the record carries information which is program readable to some data management or analysis system. The file's producer must indicate in the documentation which characters are used to mark data for which systems.

This strategy has been chosen in order to minimize the number of semantic elements which require standardization. Designers of data systems may develop their own standards for the transfer of program readable information, but these standards are transparent to the Interchange standard. There is little to be gained by establishing any common list of data system identifiers to use in column 7 of the documentation record. It is easy for a user to look in the file documentation, while it is difficult to maintain and update a list of standard identifiers.

Variable description record. The variable description record stores a variable's number, name, location and width, label, level in the file, and whether the variable is to be considered a number or a character string. Since the recording of missing data values may require more space than will be available on the variable description record, all missing data information will be placed on a separate dictionary record.

The format for the variable description record is:

Column	Information
1	Record type: V
2-6	Variable number. The variable number will be the basic data identifier in the Interchange file.

7-9 Relation number.
A relation is a set of variables referring to the same type of observation. Examples of relations are information about a single household, or information on a single individual in one wave of a panel study. Rectangular files will have only one relation.

10-17 Variable name.
This field is designed to carry variable identifiers generated by systems which refer to variables by alphabetic names. The field can also be used to hold an OSIRIS reference number, which also serves as a variable name independent of the ordering of variables in the file.

18-19 Record number.
This field allows the support of data on cards or other unit records. Interchange dictionaries and data files stored on disk and tape will usually have only one record per observation. However, when a file is stored on cards, this field will indicate the the sequence order of the card carrying the variable. A blank in this field should probably be allowed to indicate that the observation is stored on one and only one record.

20-24 Location.
This field records the location of the high order character in the variable, counted from the left edge of the record. The count includes all linking information required to associate a record with records in other groups, but does not include the binary length field which is part of the format VB record. Thus, the location field will give an accurate account of the variable's position once the record has been deblocked.

25-28 Width.
This field records the width of the variable in characters.

29 Field type.
This field is a "0" for numeric data, a "1" for purely alphabetic data, and a "2" for variables which are numeric in some observations, and alphabetic in other observations. The latter case can occur in systems which generate alphabetic missing data codes for numeric variables.

30-31 Number of decimal places.
This field has two columns to accommodate very large numbers, and numbers with a negative number of decimal places. The latter case can occur where income is being stored in hundreds of dollars. Since all data are in character, rather than in binary form, very large and very small numbers may be written into the data file in E-format.

32 Spare place for later expansion.

33-72 Variable label.
The maximum length of a variable label is forty characters. An interpolated set of documentation records may be used to extend the variable labeling information, but only the forty characters on the variable description record will be considered program readable for Interchange purposes.

73-75 File identification.

76-80 Sequence number.

Missing data record. The missing data record contains missing data specifications for the variable in columns 7-72. Since missing data specifications vary widely among systems, it seems best to allow the greatest possible flexibility in the specification of missing data. The most general way of specifying missing data would be as a Boolean expression describing which numbers and character strings will be used to represent missing values.

In this instance, the full Boolean format can be abbreviated. The "or" connective can be implied by a simple sequence of values. The statement, "If V is missing, then V equals 7 or V equals 8 or V equals 9," is well defined by "7 8 9". "If V is missing" is implied by the missing data record itself. The phrase "[or] V equals" can be used as the default meaning of a delimiter. If a fuller representation is desired, the missing data example above can be rendered as "7 OR 8 OR 9". If the "OR" default is used, then the necessary connectives are "AND", "(", ")", and "'". The relational operators, "LT", "LE", "EQ", "NE", "GE", and "GT" complete the set of primitives needed to form a missing data language. (It is not clear that "NE" has any real use in such a language.)

Examples of missing data codes in most systems are easy to express in this language. Some examples are:

```
SPSS     "77 88 99"
OSIRIS   "99 GE 77"
PICKLE   "LT 10 GT 90"
SAS      ".A .B .C .D .E .F .G .H"
```

Things are relatively simple when missing data codes are entirely numerical. However, there is some question of whether ranges of character strings ought to be allowed in missing data expressions. To do so implies that there is a common collating order. If this is the case, then the ASCII collating order could be specified as the order underlying such expressions as "(GE .A AND LE .Z)", which would neatly express all of the SAS internal missing data codes. The use of collating order ranges for alphabetic missing data codes is attractive, but may violate the primary criterion of relentless simplicity which underlies the design of the Interchange file.

One way to express a set of universal missing data codes might be to specify all missing data codes for the file with a single set of missing data records with a variable number of 00000. Thus a SAS data set in

Interchange format would have a single set of missing data records spelling out the 26 SAS missing data codes. This single set of records would apply to the entire file. The description of global blank treatment follows immediately from this procedure. Global treatment of blanks as missing data is indicated by a missing data record with a variable number of 00000 which carries a blank between primes. If desired, the file standard can be written so that variables with local missing data declarations are exempted from any global missing data declaration.

Category label record. The category label record contains a value for a categorical or discrete variable, a label of up to 20 characters, and an optional frequency count for the category. It seems simplest to have a single category on a card, with the first character string in the field interpreted as the code value, the second string interpreted as the label, and the third string interpreted as the frequency count. This set of conventions will allow the correct interpretation of,

2 FEMALE

and of

2 FEMALE 500

if blanks are allowed as string delimiters. However, additional delimiters are required for the proper interpretation of such label data as

3 FATHER'S HOUSE 405

It seems best to require that all labels containing blanks be enclosed in primes so that frequencies can be added without having to reformat the record.

Structure Description techniques

The Interchange dictionary describes each of the several types of records contained in the data file and the structural relation between record types. The structure definition facilities of the Interchange file will support rectangular files, hierarchical files, generalized network data bases of the CODASYL type, and relational data bases.

Rectangular files

Rectangular data files contain only a single type of record, and thus require no explicit structure definition. Where an interchange file is being used to store data from a DBMS or a set of files, a rectangular file may be stored as a single relation.

Hierarchical Files

Hierarchical, or tree structured, files are the most complex data structures which can be processed sequentially. It thus makes sense to provide information which will allow the sorting of the file into meaningful order. The most robust linkage method is to give each record a complete set of upward pointers. For example, consider Figure 1, which represents a file having six types of records in three hierarchical levels. Every record in this Interchange file will carry six identification numbers, one for each type of record. Each record will carry the identification variable of all records under which it can be structured. Thus the record of a child will carry numbers identifying the records of its family, neighborhood, class, and school. The child's record will carry missing data in the field carrying a "parent" pointer, since children are not subordinated to parents.

All possible linkages may be represented by such sets of upward pointers. The assignment of levels does not always imply that records at a lower level are disaggregations of records at a higher level. In some cases the specification of levels is simply to resolve which record is pointing at which.

Although the set of pointers in Figure 2 is implied by the linkages in Figure 1, it would be difficult to infer those linkages solely from analysis of the pointers, since to do so would require reading the entire file. The structure can be inferred from the information that parents and children never point to each other but both point to families, and that families point to neighborhoods, while neighborhoods do not point anywhere.

The structure definition, however, should be provided explicitly, so that both the user and the file importer know what to do with the data. Thus the Interchange dictionary should have an explicit structure definition record as well as a set of record definitions giving pointer information. The person creating an Interchange file must choose an identification variable for each type of record. People should, in general, be discouraged from creating records without identification variables. If no variable is appropriate as an identifier, then the file exporter should supply an arbitrary sequence number for each record. The sequence number need not be unique within the file, but only within the level at which the record enters the structure. For example, the records of children in Figure 1 may carry unique

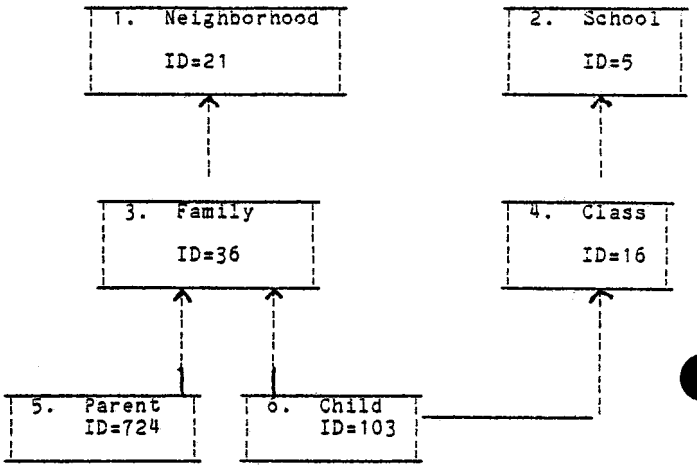


Figure 1: A hypothetical data structure.

	Pointer #						
	0	1	2	3	4	5	6
Record Type 1	1	21	-	--	--	---	---
Record Type 2	2	--	5	--	--	---	---
Record Type 3	3	21	-	36	--	---	---
Record Type 4	4	--	5	--	16	---	---
Record Type 5	5	21	-	36	--	724	---
Record Type 6	6	21	5	36	16	---	103

Figure 2: Pointer array for the data structure in Figure 1. Pointer 0 is the record type.

identification numbers, but if they do not, a simple sequence number within each family will suffice. Each of the N types of records in a tree structure will be prefixed by N + 1 identification variables, consisting of a record type identifier and N pointers, one for each type of record in the file. Where two records are connected by more than one link, then more than one pointer will be required. Hopefully, people will be sparing in their use of multiple identification fields.

It should be noted that at least one pointer on each record is a simple duplication of one of the variables in the record. The duplication is justified in order that the syntax of the pointer variables be completely under the control of the file exporter, and thus absolutely canonical in the Interchange format. The user may use almost anything as a missing data indicator in the data portion of the record, but when that identifier is copied into the pointer section, it will be subject to conventions specified in the Interchange data standard. Such conventions should require that pointers have only numeric values, that there be no blanks in pointers, and that a particular type of missing data indicator be used. (See Tsichritizis and Lochovsky, 1976, for a fuller exposition of hierarchical data bases.)

Network Data Bases

A network data base is a generalization of the hierarchical file. In the hierarchical file, records are related to each other in only one way, by being subordinate or superordinate to each other. In the example above, children are subordinate to families, who are subordinate to neighborhoods.

Suppose that the file contained a set of records each of which held information on a hobby, and that each "child" record contained a variable called "favorite hobby". This variable is a link between the child and information about his or her favorite hobby. Thus, the "hobby" and the "child" records need share no sorting information, but their linkage must still be documented. The linkage is documented in the structure definition by naming it and by indicating the variables which link the two records. However, there is no way in which hobby records can be sorted into the hierarchy of neighborhood, family, and child. (See Taylor and Frank, 1976, for a fuller exposition of network data bases.)

Relational Data Bases

Relational data bases are constructed from rectangular files which have unique identification variables. Relational data bases have advantages in being uniquely simple and general. Their present disadvantage is that there are at present relatively few implementations of such data base systems. Each "observation" in a relational data base is a rectangular file with a unique identification variable. The relational data base is simple because it uses no system of pointers. "Observations" or "relations" as they are called in data base management terminology, are linked by a process of sorting and merging on the basis of identification numbers.

As long as the proper conventions of uniqueness of identification variable are maintained, the interchange file will support relational data bases in the form of heterogeneous records with no explicit hierarchical relation. Each relation a relational data base is simply assigned a relation number in the interchange file. (For a fuller exposition of relational data bases, see Chamberlin, 1976, and Teitel, 1977.)

Matrices and Tables

The interchange file stores matrices and multidimensional tables in straightforward fashion. A matrix or two dimensional table may be stored as a rectangular file. A table of higher dimensionality may be stored in either "row" or "cell" fashion. An n-dimensional table stored in "row" fashion is treated as an n-1 level hierarchical file with a single record type. Each record in a "row" file is a vector from the table, labeled with its coordinates in the table.

A 3 x 4 x 10 table could be stored as a file of 12 records, each of which contained 10 variables. Each or the records would be treated as the third level of a hierarchical file, and would have pointers indicating the row and plane of the table to which it belongs. Such a file could easily be sorted by row and plane. However, a sort by column would have to be performed by reformating the records.

A seemingly more clumsy, but preferable way of storing a table is in "cell" form, in which each cell of the table is labeled with its coordinates. The coordinates are in canonical form as pointers rather than variables, and are thus provided by the exporting program, rather than by the user. Although a "cell" file might appear quite bulky in relation to the original table, it provides the advantages of being totally open in format and of being possible to manipulate without reformatting records.

Structure and Record Definition Records

Record definition record. The record definition records constitute a dictionary for the rectangular subfile formed by the type identifier and vector of pointers. The format for the record description record is:

Column — Information

1 Record type: R.

2-6 Variable number.
The variable number has no direct application to the record definition record but is used solely for sequencing in the file. Thus, any variable number less than the smallest variable number can be used. Users should probably be encouraged to number variables in a way which helps identify their record type, such as having variables in record type 5 begin with 501.

7-9 Relation number.
Several options are available for the formatting of relation numbers. One option is that the data Interchange specification require that the first three columns of every data record be a record type identifier. The second alternative is that the location of the record type identifier be inferred as everything ahead of the first pointer field. Thus, if the pointer for the lowest record type begins in column three, columns one and two of the record are assumed to be the record type. A third, and perhaps the most suitable alternative, is that the definition of record type zero indicate the location of the record type indicator.

10-11 Level.

12-31 Name.

32-36 Pointer location.

37 Pointer width.

38-46 Pointer missing data value.

47-55 Pointer inappropriate value.
It has been suggested that separate missing data and inappropriate codes are not needed for pointers. A missing data code in a pointer to a higher level can be interpreted as actual missing data, while a missing data code in a pointer to the same level or to a lower level can be inferred as inappropriate. It has not yet been decided whether or not inappropriateness should be explicit or inferred.

56-60 Pointer variable number.
This field indicates the variable number in the record type which has been used as the pointer. A variable number of zero indicates that the file exporter produced an arbitrary sequence number for the record.

61-65 Number of variables in the record.

66-70 Aggregate record length.
These two fields would be helpful in allowing the importing program to allocate work space for reformatting the file. However, they may require two passes through the file to create the dictionary and might be omitted from the record definition record. Further discussion of whether or not to include them is necessary.

73-75 File identification.

76-80 Sequence number.

Structure definition record. The structure definition provides an explicit indication of the links between record types. The structure definition consists of a set of free format expressions indicating the equivalence between pointers in different relations. The format for the structure description record is:

Column — Information

1 Record type: S.

2-6 Variable number.
The variable number has no direct application to the record definition record but is used solely for sequencing in the file. Thus, any

variable number less than the smallest variable number can be used.

7-72 Structure definition expressions. Free format expressions showing the logical structure of the file and the variables linking different relations.

73-75 File identification

76-80 Sequence number.

A suggested syntax for structure description expressions is:

<name>:<rectype>(<var#>)=<rectype>(<var#>)

In the case of hierarchical records, it would be nice to require that the direction of the expression go from lower level to higher level in order that the hierarchy in the file be inferrable without reference to the level numbers contained in the record definition records. Hierarchical files do not need explicit names for pointer relationships. However, names are necessary to clarify the relations between the records of a generalized network data base. The structure of the file in Figure 1 could be indicated (using arbitrary variable numbers within record types) as:

5(2)=3(1) 6(2)=3(1) HOME_ROOM:6(3)=4(1)
3(2)=1(1) 4(2)=2(1)

This structure definition allows both the user and the importing program to recover the original structure of the file.

Entry definitions. Following the OSIRIS convention, an entry is defined as the rectangularized file actually read and analyzed by a program. The OSIRIS structured file carries with it a default entry definition which is used in the absence of any specification by the user. There is some question as to whether the Interchange file should carry a default entry definition with its dictionary. If the importing system uses a hierarchical file, then the importer could simply transform the Interchange file into an esoteric hierarchical file. However, it can be expected that many importing systems will not support hierarchical files, and that the file must therefore be rectangularized. Perhaps the most reasonable course is to include a verbal summary of some entry definition and leave the actual construction of the entry to the user and the file importer.

Data

Perhaps the only restriction on the data is that they be in the form of printing ASCII or EBCDIC characters. It would be nice to require that data be written without leading blanks, but considering the number of FORTRAN programs which will be used to produce data files, it is unlikely that this restriction would be very popular.

Interchange File Creation and Conversion

Manual Creation

Proper design of the Interchange dictionary will allow many Interchange files to be constructed without the use of special programs. Rectangular files will require the addition of an observation identifier, something which should probably be there in any case. Once such a data file has been produced, a valid Interchange dictionary can be produced by hand.

File Conversion Programs

In order for the Interchange file to succeed, statistical systems must have facilities for converting their own esoteric files to and from Interchange format. File importers will probably need special care in their design, since they must be capable of correcting the file producers' deviations from good practice. Importers will probably require not only extensive recoding techniques for converting such things as missing data codes, but also reasonably powerful text editing techniques in systems which will not support the long labels of the Interchange dictionary. In the long run, it would be far better to increase the labeling capabilities of other systems to the SPSS standard, than to degrade one of that system's most pleasant and useful features. The design of an Interchange file importer for each statistical system is a problem whose difficulty should not be minimized. Hopefully, much of the work of civilizing files which violate rules of good practice will be done by data archives.

The task of designing a file exporter seems somewhat simpler than that of designing an importer. The Interchange dictionary can be written from the system's esoteric dictionary, and the pointer section of the data records written without much difficulty.

Machine Readable Documentation

At present, the development of machine readable code books considerably lags the present state of computer text processing. Most code books are simple transcriptions of paper code books to punched card for easy transmission with the data. The OSIRIS code book, the most highly developed of machine readable code books, is basically a primitive form of document processor manuscript. OSIRIS code books are laborious to prepare and difficult to edit. Few users employ the subsetting facilities of the OSIRIS system, while even fewer ever edit, expand, or create new OSIRIS code books.

The full data Interchange file should probably include a machine readable code book. Code book information can be carried on the documentation records in literal form, and these records can even be subsetted as the file is broken into subsets. However, transmission and storage of code book information in literal form loses most of the flexibility afforded by computer document

processor systems. Code book information stored as a document processor manuscript can be easily edited, subsetted and modified. In addition the document processor will provide such features as automatic resolution of table and variable numbers and an automatic table of contents and cross reference.

Future work on the data Interchange file should include the selection of a document processing language. In the meantime, documentation on the Interchange file should probably be stored in literal form.

GLOSSARY

This glossary is intended to clarify certain terms which are used in new or unusual ways in this paper. It is not meant to be in any sense a complete glossary of terms relating to the Interchange standard.

Data set. A file or set of files containing complete information on a set of self-described data. An SPSS data set consists of one file, while an OSIRIS data set can consist of two or three files.

Dictionary. A program readable set of information describing a machine readable data file.

Entry. The data vector created from a hierarchical file which is actually read and analyzed by a statistical program.

Esoteric file. A file which cannot be interpreted with simple printed dumps and read by simple FORTRAN style format statements. Esoteric files must be read by specially designed software. SAS and SPSS files are both esoteric.

Exoteric file. A file which can be interpreted with character format dumps and which requires only a simple format statement for interpretation. Card image files are exoteric.

Exporter. A program or subprogram built into a data analysis system which generates Interchange data sets from the system's native data set.

File. A set of machine readable data organized as a unit with respect to a computer system. A file need not be coterminous with a data set. For example, several SAS data sets can occupy a single IBM file, an SPSS data set is coterminous with an IBM file, while an OSIRIS data set requires two IBM files.

Importer. A program or subprogram built into a data management and analysis system for converting Interchange data sets into the system's native data format.

Interchange data set. A dictionary file and data file constructed according to the standards outlined in this paper and agreed on by the working group.

Literal text. Text which is printed exactly as it is stored on the machine readable medium without reformatting.

Machine readable. Information stored on punched cards or magnetic media which can be interpreted by a computer. Machine readable data, e.g., literal text, is not necessarily in a form which can be interpreted by processing programs and should be distingushed from program readable data.

Pointer. The vector of identification variables prefixed to each Interchange format data record.

Program readable. Machine readable data in a form suitable for interpretation and processing by a computer program. For example a set of keywords punched on cards are both machine readable and program readable, while a comment statement is merely machine readable.

REFERENCES

Barr, J., and Goodnight, J. SAS progress report. Paper presented at the SAS Users' Group meeting, Orlando, FL, January, 1976.

Buhler, R. The P-STAT system. Pp. 283-286 in Proceedings of Computer Science and Statistics: 7th Annual Symposium on the Interface, Iowa State University, 1973.

Chamberlin, D. G. Relational data base management systems. Computing Surveys. (8),43-66.

Dixon, W. J. (Ed.) BMD Biomedical Computer Programs. Berkeley, CA: University of California, 1975.

Meyers, E. D. Jr. Project IMPRESS: Time-sharing in the social sciences. AFIPS - Conference Proceedings. (34), 673-680, 1969.

University of Michigan. OSIRIS III: Volume I, system and program description. Ann Arbor, MI: Author, 1976.

Nie, N. H., Hull, C. H., Jenkins, J. G., Steinbrenner, K., and Bent, D. SPSS: Statistical Package for the Social Sciences. New York, McGraw-Hill, 1975.

Roistacher, R. C. The data interchange file: A first report. Center for Advanced Computation. (CAC Document No. 207), 1976.

Taylor, R. W. and Frank, R. L. CODASYL data base management systems. Computing Surveys. (8), 67-104, 1976.

Teitel, R. F. Data base concepts for social science computing. Proceedings of Computer Science and Statistics: 9th Annual Symposium on the Interface. Harvard University, 1976.

Tsichritizis, D. C., and Lochovsky, F. H. Hierarchical data-base management. Computing Surveys. (8), 105-124, 1976.

WORKSHOP 9

COMBINATORIAL COMPUTING IN STATISTICS

Chair: Joseph B. Kadane, Carnegie-Mellon University

THE WHYS AND WHEREFORES OF ALGORITHM DESIGN
(An Introduction for the Skeptic)

Jon Louis Bentley
Departments of Computer Science and Mathematics
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

ABSTRACT

The field of "Algorithm Design" abounds with beautiful theorems for the theoretician and tools which can save kilobucks for the engineer. This survey shows how an understanding of fundamental algorithmic issues can benefit anyone involved with computing.

1. INTRODUCTION

"Algorithm design--that's the field where people *talk* about programs and *prove theorems* about programs instead of *writing and debugging* programs." I've heard statements along those lines from applications programmers and academicians alike. But I've also heard, "No! Proper algorithm design has helped us to save kilobucks at our installation every month." In this paper we will investigate the field of algorithm design (which also is known as "Analysis of Algorithms" and "Concrete Computational Complexity", among other names) and better equip the reader to judge the field for himself.

I trust that anyone who has even the slightest love for mathematics burning somewhere inside his heart (however deeply), will continue to read this paper to see how mathematical tools can be applied to the problems of programming. But for the rest of the readers (whose interest in mathematics was probably squelched in freshman calculus) I'll have to offer the same bait that drew me into this field. I can trace my interest in the design of efficient algorithms to the time when I was a Business Data Processing programmer and had just finished reading an introductory text on "Data Structures". A colleague of mine had just had his job cancelled--the operators had estimated (by counting the turning rate of the tapes) that it would take about three hours to process his one reel of data. The program itself was fairly short and a quick glance told us that all of the time was spent in scanning a one thousand element table. I suggested that instead of scanning we try a new-fangled technique I had just read about--binary search. We did, and the modified program processed the reel of tape in *five minutes* (and spent almost all of its time waiting for the tape!). Around that same time I was asked to help another programmer who had already spent one month of time and produced about a thousand cards of code for a particular program. A simple change in data structure and less than a week's work (starting over from scratch) allowed us to redo the program in less than two hundred lines of code. The resulting program was faster than the original would have been, used far less code, and was much easier to understand. So even if you have no aesthetic interest in algorithm design (yet), please read on--the practical benefits alone can sometimes be rewarding enough!

Throughout this paper we will refer only to the *discrete* aspects of algorithm design, or to the design of discrete algorithms, if you will. We will not even mention numeric problems such as stability, truncation error, error propagation and other issues that are in the domain of numerical analysts. It turns out that even with this restriction, we still include some very numeric problems, such as the manipulation of sparse matrices (in which almost all elements are zero) and the Fast Fourier Transform.

A number of survey papers on the field of algorithm design have appeared recently. Hopcroft [1974] and Tarjan [1977] both give a broad and thorough picture of the field. Weide's [1977] survey concentrates on the techniques used for analyzing discrete algorithms, and accomplishes that task expertly. For those who are skeptical of sweeping surveys and prefer to see a couple of problems examined in detail, Knuth's introductions [1971, 1977] will prove enlightening and fascinating. And if one is ready to become a serious student of the field, the canonical texts are provided by Aho, Hopcroft and Ullman [1974] (a one-semester, graduate level introduction) and

Knuth [1968, 1969, 1973] who has completed three volumes of his seven volume *definitive* work on computer algorithms.

In this paper I will try to supplement those works by providing a broad survey for the uninitiate. Although I hope that the beauty of this field will not go unnoticed, I will try to emphasize its utility to the practicing computer user. My goal in this paper will be to communicate the *flavor* of the field, and I therefore abandon any pretenses as to the completeness or thoroughness of this study. I urge the reader to keep in mind that the contents of this paper are not the established paradigms of the field, but merely one man's view of the currents in his area. The bibliography has been kept exceptionally short for the sake of brevity; both Tarjan [1977] and Weide [1977] contain excellent bibliographies for those interested.

This paper is divided into five sections, which should probably be read in order. In Section 2 we will examine four problems and some algorithms for solving them. Having examined those concrete examples we turn to a systematic view of the field in Section 3. In Section 4 we will mention some of the current directions in which the field is now moving. Finally we tie together the main points of this paper in Section 5.

2. EXAMPLES OF FAST ALGORITHMS

Sweeping generalizations without supporting examples are often content-free, so before we go on to our sweeping generalizations in Section 3 we will study a few examples of fast algorithms. For each example we will specify a problem, mention some of its real-world applications, give an algorithm to solve the problem, analyze the efficiency of the algorithm, and then discuss interesting issues which have surfaced. We will study the "subset testing" problem of Section 2.1 in a fair amount of depth and then treat the other three problems at a more superficial level. After discussing these examples, and before we move on to the statements about the field of algorithm design in Section 3, we will summarize what all of our work bought us in Section 2.5.

But first a word on why we are examining these particular problems. The subset testing problem will raise a number of familiar issues and should cover some old ground for almost everyone; it also gives us a nice illustration of the tremendous time savings achievable with proper algorithms. The substring searching problem of Section 2.2 provides an extremely interesting blend of theory and practice. The Fast Fourier Transform of Section 2.3 is known to many, uses some important algorithmic techniques, and is eminently practical. In Section 2.4 we examine a very old problem (matrix multiplication) and a recent and remarkably counter-intuitive solution; we will also see some mysterious relations among problems that appear to be at opposite ends of any spectrum!

2.1 Subset Testing

Given a set A (of size n) and a set B (of size $m \le n$), is B a subset of A?[1] This "subset testing" problem can be stated as a programming exercise: given an array A[1:n] and B[1:m], both of (say) 32-bit words, is every word in B also in A? Disguised versions of this problem arise in many contexts: A could be an employee master file, B a list of weekly transactions, and we want to find a master-file record for each weekly transaction. Or A might be a table of real numbers x and have an associated table S which contains sine x, then B would be a set of x values at which the sine function is to be evaluated. Although this problem does have some practical application, that is not our main motivation for examining it here. We will see that it leads to many of the basic issues in sorting and searching, and points to inter-relationships between those problems. We will also get an exposure to some of the common methods of algorithm design.

We will examine three ways of solving this problem. In order to compare the methods we will find the running time of each by counting the number of comparisons between elements each requires. This will not give the exact running time, but should give us a fairly robust idea as to the relative merit of each. (Don't worry; we'll return to this issue later!)

Brute Force

The simplest way to accomplish this task is to compare every element in B to each of the elements of A until either its equal is found or we have examined all of A and determined that it has no equal in A (in which case B is not A's subset); this approach gives a simple, two-loop program. If B is indeed contained in A, then each scan for an element that is B's mate in A takes n/2 comparisons on the average (you have to look halfway down the list). Since there are m such scans made, the total number of comparisons made by this program is about m(n/2). So if m is very close to the size of n, then we will make about $n^2/2$ comparisons on the average[2]. Although this algorithm is exceptionally simple to understand and to code, its slow running time might prohibit its use in certain applications. We will now turn our attention to an even faster algorithm.

Sorting

If I were to give you a randomly ordered list of phone numbers B (say a list of phone numbers in a town) and another randomly ordered list A (say all phone numbers in the county) and asked you to check whether B was a subset of A (make sure every town phone number is included in the county list), then you might use the brute-force algorithm we just discussed. If, however, I handed you a town phone book and a county phone book and asked you to perform the same task, then your job would be a lot easier. Since the two phone books are already sorted (by name) we can just scan through the two books together, insuring that the county book contains all the town names. This of course immediately gives us another algorithm for subset testing: sort A, sort B, then scan through the two, checking. To analyze the run time of this strategy we observe that the scan will take about m+n comparisons, and we heard somewhere that you can sort a list of size n in about $n \log_2 n$ comparisons, so the total running time is $(n \log_2 n) + (m \log_2 m) + m+n$ comparisons.

We could just pull a sorting routine out of thin air, but it

1 This problem is discussed by Knuth [1973, p. 391].

2 We won't try to analyze the case that B is not a subset of A; to do so we would have to say exactly how it is not a subset, and that is very dependent on the particular problem!

isn't much more difficult to describe one called Mergesort. The basic operation of Mergesort is merging two sorted lists[3] of numbers, say X and Y. To do this we compare the first element of X with the first element of Y and give the smallest as the first element of the new list, deleting it from its source. We repeat this remove-the-smallest step until both X and Y are empty. Since we used one comparison for each step, if there were a total of m elements in X and Y, we will have used about m comparisons[4]. We can now use this tool of merging to Mergesort a set S of n elements. We start by viewing S as a set of n sorted one-element lists. We then merge adjacent pairs of one-element lists, giving n/2 2-element sorted lists. The next step is to merge adjacent pairs of those lists giving n/4 4-element lists, and the process continues. After $\log_2 n$ iterations we have one sorted n-element list, and our task is complete. To analyze this we note that we use about n comparisons for the merges at each of the $\log_2 n$ iterations, so the total number of comparisons used is the promised $n \log_2 n$.

We can view Mergesort from the vantage point of recursion and achieve a totally different perspective on the same algorithm, which might reinforce our intuition and ease our analysis. To Mergesort a set S of size n we divide S into two sets X and Y (each of size n/2), sort X and Y by recursively calling Mergesort, and then merge the sorted lists together. This algorithm will have exactly the same effect as our iterative version, and (at least for those experienced in "recursive thinking") is easier to conceptualize. We can also use this viewpoint to analyze the algorithm. Since it takes no comparisons to sort a one-element list, from our algorithm we know that if T(n) is the number of comparisons required to sort an n-element list then it satisfies the recurrence relation

$$T(1) = 0,$$
$$T(n) = 2T(n/2) + n$$

which has solution $T(n) = n \log_2 n$. (The recursive part reflects the fact that to sort an n-element list we must sort two (n/2)-element lists and then use n comparisons to merge those.)

We have now shown how to solve the subset problem with $n(\log_2 n + 1) + m(\log_2 m + 1)$ comparisons. If m is about the same size as n then our algorithm takes $2n \log_2 n$ comparisons. Can we do better?

Hashing

Our introspection as to how we would solve the phone book problem led us to discover an interesting sorting approach to the subset problem; if we rephrase the phone book problem our "human" approach will lead us to an even faster subset algorithm. Suppose that the county phone book (A) was sorted and the town phone list (B) was not; to insure that A contains B we can "look up" in A each number in B by the name of the subscriber. For each of the m elements in B we would do a "binary search"[5] among the n elements of A. It is not hard to see that a binary search in an n-element sorted table takes at most $\log_2 n$ comparisons, so this algorithm is easily analyzed. We therefore have a searching solution to the subset problem: store the elements of A in a table, then for each element of B insure that it is in the table.

Although binary search is the best searching method for many problems, there is another searching strategy even more appropriate for this problem: hashing. Using hashing we can store an element in a table or check to see if an element is already in a table in about two comparisons, on the average[6]. With this approach we will be able to do subset testing in $2n + 2m$ comparisons--2n to store A then 2m to look up each element of B. To store the n elements of A we will have to allocate a *hash table* which is an array of length (3/2)n.[7] We then store the elements of A in the table one-by-one by the use of a *hash function*. This function maps a data value into an integer in the bounds of the hash table. If that position in the hash table is empty, fine: we insert the element. If the position was occupied, however, we have a *collision*, and must employ a *collision resolution strategy*, such as scanning up the elements of the array until a free position is found. Analysis has shown that a proper collision resolution strategy allows one to find an empty spot very quickly (say, in two comparisons). When an empty spot is finally found the element is inserted. After inserting all of A's elements into the table we then look up all of B's elements. For any particular element we calculate its hash function and look in that position. If that position is empty then it is not in A; otherwise we must employ the same collision resolution strategy to see where it should be. The technique of hashing is something that a person would never use in his searching (we are much better at comparing things and then looking in one of two directions than at calculating weird hash functions), but it leads to a very efficient algorithm. If m is about the same size as n then the hashing approach uses only about 4n comparisons (on the average) to do subset testing.

Summary

The subset testing problem is stated very simply but has led us straight to some of the fundamental problems of algorithm design. We very quickly arrived at searching--the scan of the brute force algorithm is just a naive search. From there we moved to sorting, then to binary search, and finally to hashing, which introduced us to a non-obvious data structure (the hash table).[8] The approaches that we used to solve these problems are some of the fundamental tools of algorithm designers. We have also touched on a number of interesting aspects of algorithmic problems such as time and space analyses and worst-case versus expected-time analysis. We will study these issues further in Section 3.

But what has all this gained us? We certainly have a more definite understanding of some of the fundamental computational problems involved, but does it make any

3 We won't worry right now about their implementation--they could be either arrays or linked lists with pointers.

4 Actually at most m-1, so our results are a little pessimistic.

5 A binary search for a name in a phone book first compares that name to the middle name in the book. If that name is less than the middle we restrict our search to the first half of the book, otherwise we search the last half, and so on.

6 For pessimists, however, the worst case of hashing is horrible--we might have to look at all of the elements in the table.

7 We could use a smaller array; (1.1)n would probably work almost as well.

8 For a more thorough examination of searching the reader should see Knuth's [1977] survey.

difference in practice? To answer this question let's assume that we are writing a program for subset testing where A and B both contain one million elements, and for the sake of argument assume that one comparison takes one microsecond of computer time. By these assumptions, the $n^2/2$ comparisons required by brute force translates to 138 hours (or a little shy of six days) of machine time; the $2n \log_2 n$ for sorting will give 40 seconds; and the 4n of hashing will yield 4 seconds. Although we haven't calculated all the costs of implementation, this example shows how sometimes a simple analysis is all you need to make an informed choice!

2.2 Substring Searching

Does a given *string* contain a specified substring *pattern*, and if so, where? This is the substring searching problem. This problem is familiar to most who have used computer text editors; as I sat down to type this paragraph I told the editor to find the substring "2.2" in my text file so I would know where to insert this text![9] This same operation is used in information retrieval systems as they try to identify abstracts which contain certain keywords. Similar problems are encountered in many macroprocessors and text formatters.

It is not hard to write a program to solve this problem. We first hold *pattern*'s leftmost character under *string*'s leftmost character and start comparing. If all the characters of *pattern* match the characters above them, fine--we have found the substring in position 1. If we find a mismatch we slide *pattern* over one and do the same thing again. This continues until we either find a match or come to the end of the *string*. The worst-case behavior of this algorithm is very slow--for each of the n positions of *string* we might have to compare all m positions of *pattern*. Thus in the worst case we might have to make mn comparisons. *Strings* and *patterns* that realize this worst-case behavior are fairly pathological and the performance of this algorithm in practice is fairly good, but the question still haunts us--can we give an algorithm that will always do better?

Knuth, Morris and Pratt [1977] give an algorithm that beats that bound. They preprocess *pattern* into a data structure that represents a program; that program then looks for *pattern* in *string*. Preprocessing *pattern* by their algorithm takes only m operations (where m is the length of *pattern*) and the "program" they produce looks at each character of *string* only once, so the total running time of their algorithm is proportional to m+n. (Of course if the pattern is in the string in position i, then their algorithm takes time proportional to i+m.) This result is exceptionally interesting from a theoretical viewpoint, and also provides a faster substring searching algorithm in practice.

Boyer and Moore [1977] recently used the basic idea of the Knuth, Morris and Pratt algorithm to give an even faster method of substring searching. Their method has the same worst-case performance (proportional to m+n), but is somewhat faster on the average. They accomplish this by making it unnecessary to examine every element of *string*. They have implemented their algorithm on a PDP-10 so efficiently that when *string* contains typical English text and *pattern* is a five letter word in *string*, the number of PDP-10 instructions executed *is less than i+n*. This is at least an order of magnitude faster than the naive algorithm.

The history of the substring searching problem provides an interesting insight into the relation of theory and practice in Computer Science. Knuth relates that he was led to his discovery of the algorithm by the use of a machine from automata theory called the "two-way deterministic pushdown automaton". The easiest way to understand the fast algorithms is through the use of another abstract machine called the "non-deterministic finite state automaton". It is noteworthy that in this one problem we talk about such diverse ideas as abstract automata and PDP-10 instructions, with a lot of difficult combinatorial analysis in between!

2.3 The Fast Fourier Transform

The Fourier Transform is often studied in mathematics and engineering. It can be viewed in a number of ways, such as transforming a function from the "time domain" into the "frequency domain" or as the decomposition of a function into its "sinusoidal components". The continuous Fourier Transform has a discrete counterpart, which calls for applying an operation to one set of n reals yielding a "transformed" set of n reals. This problem has applications in signal processing, interpolation methods, and many discrete problems.

The naive algorithm for computing the Fourier Transform of n reals requires approximately n^2 arithmetic operations. The Fast Fourier Transform (usually attributed to Cooley and Tukey) accomplishes this task in approximately $n \log_2 n$ arithmetics. It does this by doing about n arithmetics on each of $\log_2 n$ levels; in this sense it is quite similar to the Mergesort algorithm of Section 2.1. There are many different expositions of the algorithm; see Aho, Hopcroft and Ullman [1974] or Borodin and Munro [1975]. (It is interesting to note that in addition to being faster to compute, many of the numeric properties of the FFT are better than those of the naive transform.)

The Fast Fourier Transform has had a substantial impact on computing. It forms the backbone of many "numeric" programs. The FFT has been used in diverse fields to find hidden periodicities of a stationary time series. In signal processing it is used in filters to remove noise from signals and eradicate blurring in digital pictures. It is used in numerical analysis for the interpolation and convolution of functions. Applications of the FFT in such diverse areas as electrical engineering, acoustics, geophysics, medicine, economics, and psychology are listed in Bullinger [1975, Section 1.5]. Many special-purpose processors have been built which implement this algorithm; some of those are multiprocessors which operate in parallel. The FFT is also widely used in the design of "discrete" algorithms. It is the primary tool in many algorithms which operate on polynomials, performing such operations as multiplication, division, evaluation and interpolation. Not surprisingly, it is also employed in some of the fastest known algorithms for operating on very long integers (such as multiplying two one-million bit integers).

2.4 Matrix Multiplication

One of the most common ways of representing many different kinds of data is in a matrix, and one of the most

9 The text editor I use looks at my file as one long string of text, sprinkled with characters representing "carriage return".

common operations on matrices is multiplication. How hard is it to multiply two n x n matrices? Using the standard high school method takes about 2n arithmetic operations to calculate each of the n^2 elements of the product matrix, so the total amount of time required by that algorithm is proportional to n^3. People have been multiplying matrices by this method for centuries. Surely this must be the best possible way to multiply matrices--our intuition tells us that we just can't do any better.

The high school algorithm for multiplying two-by-two matrices uses 8 multiplications and 4 additions. It is fairly counter-intuitive to learn that the product can be computed using only 7 multiplications at the cost of an increase to 15 additions. But if that is counter-intuitive, then it is absolutely mind-boggling to find that that fact alone allows us to construct an algorithm for multiplying n x n matrices that runs in less than n^3 time! This algorithm is due to Strassen [1969] and works by decomposing each n x n matrix into four (n/2) x (n/2) matrices. To find the product of the original matrices it does seven multiplications of (n/2) x (n/2) matrices and then fifteen additions on matrices of that size. Notice, however, that the cost of those additions is proportional to n^2. If we let T(n) be the time required to multiply n x n matrices, then T(n) satisfies the recurrence

$$T(n) = 7T(n/2) + O(n^2),$$
$$T(1) = 1$$

which has the solution $T(n) = O(n^{2.81})$ (where 2.81 is an approximation to $\log_2 7$). Using the naive implementation of this algorithm would probably prove less efficient than the high school algorithm until n was in the thousands; recent work, however, has shown that it can be practical when n is as small as 40. But practice aside, who can help but be amazed by the fact that we can multiply matrices faster than we thought we could?

The fast matrix multiplication algorithm provided the basis for one of the all-time great revolutions in the history of algorithm design, during which a number of "best known" algorithms were toppled from their reign. Many of these were n^3 matrix algorithms which we can now do in $O(n^{2.81})$ time; among these are matrix inversion, LUP decomposition, solving systems of linear equations, and calculating determinants. A number of problems which seemed to be totally unrelated to matrices were phrased in that language and $O(n^{2.81})$ algorithms followed for such diverse problems as finding the transitive closure of a graph, parsing context-free languages (an important problem in compilers) and finding distances between n points in Euclidean n-space. All of these algorithms stem from the fact that two-by-two matrices can be multiplied with seven multiplications! If only they had told us about things like this in freshman mathematics, instead of making us memorize integral tables!

2.5 So What?

We have now examined four cases in which proper algorithm design has led to a sophisticated algorithm which is much faster than a naive algorithm. A lot of work has been invested in developing these algorithms; what difference will all this work make in practice?

To be honest, most of the time a fast algorithm makes no difference at all. Knuth has gathered empirical evidence which shows that most of the run time of a program is spent in just three percent of the code (this should come as no shock to statisticians who know that twenty percent of the population accounts for eighty percent of the beer consumed). If the problem to be solved is not in the critical three percent of the code (as about 97 percent of the problems are!) then it makes little difference if that algorithm is fast or not. A more complicated algorithm can often be a liability rather than an asset. It will usually mean more coding and more debugging time, and can sometimes even increase the run time (when the overhead of "starting up" a fancy algorithm costs more than the time it saves).

Sometimes, however, a fast algorithm can make all the difference in the world. If the computation being performed is indeed the bottleneck in the system flow, then an algorithm of half the running time almost doubles system throughput. In many text editors the vast majorty of the time is spent in string searching; the fast algorithm of Section 2.2 can speed up many text editors by a factor of five. My experience with the searching program which I mentioned in the introduction (when we reduced the running time of a program from three hours to five minutes) is another classical example of an appropriate use for a fast algorithm. In the inner loops of computation-bound programs (those not waiting all the time for I/O), proper algorithm design is critical.

An analogy will perhaps clarify these issues. It is fairly easy to walk, it more complex to drive, and it is even more complex yet to learn to fly a modern jet airplane. Walking is the best way to get from one room of a house to another, driving is superior for getting from one town to another, and flying is hard to beat for getting from one part of the country to another. There is no "best" mode of transportation--the best mode in a particular case depends strongly on that case. For most of us the time we spend travelling in jet airplanes is very small compared to the time we spend walking--but it sure is nice to know about jets when we need them!

Although the effort of fast algorithm design only occasionally gives us large financial savings, it always gives us something of a different value--a fundamental understanding of our computational problems. This is usually reflected in cleaner programs. But even more important is the understanding of how difficult it is to compute something. After a student has spent a month or two investigating the problem of searching, he not only knows how to search fast but also why he can do it that fast and why he can't do it any faster. Such a student has learned something of the very foundations of his field.

3. A SYSTEMATIC VIEW

In Section 2 we saw a number of particular problems and a number of particular solutions; in this section we will show that there is more to the field than isolated examples. In Section 3.1 we will discuss the concepts one needs to define a computational problem, and in Section 3.2 we will use those concepts to describe the kinds of problems for which fast algorithms have been designed. In Section 3.3 we will take a peek into the algorithm designer's tool bag.

3.1 Dimensions of a Problem--A Microscopic View

The subset testing problem of Section 2.1 showed that there can be many different algorithms for solving a particular problem. In order to say which one is best in a particular application we have to know certain *dimensions* of the problem. For example, in one application we may need a subset algorithm that must be very space-efficient and have good worst-case running time; in another context we might have a lot of available space and only require good expected running time, not caring if we (infrequently) must take a lot of time. We have thus identified three dimensions of a computational problem: time analysis, space analysis, and expected vs. worst-case analysis. In this subsection we will discuss these and other dimensions of computational problems.

Time and Space Analysis

The two most important resources in real computational systems are time (CPU cycles) and space (words used), and these are therefore the two dimensions of a problem most frequently discussed. The running time was the primary subject we examined in the examples of Section 2. Most of the algorithms we examined use very little extra space after storing the inputs and outputs; the hashing algorithm of Section 2.1 was the only exception. In large computer systems huge quantities of extra space (megawords) can be had for the asking and the paying; for that reason many have ignored the space requirements of algorithms. With the rise in popularity of mini- and microprocessors with very small memories, however, space analysis is once again an extremely important issue.

Model of Computation

Throughout Section 2 we were able to make reference to the time and space requirements of various algorithms without reference to their implementation on any particular computer. Our intuitive notions were robust enough to lead to sophisticated algorithms that will certainly beat their naive competitors on any existing machine. But to analyze an algorithm in detail we must have a precise mathematical *model* of the machine on which the algorithm will run.

We could choose as our model a particular computer, such as an IBM 650 or a DEC PDP-10, and then ask how many microseconds of time or bits of storage a particular algorithm requires. There are two problems with this approach. First, we will probably be analyzing the expertise of the implementor of the algorithm more than the algorithm's intrinsic merit, and second, once we have completed such an analysis using the IBM 650 we still know very little about the algorithm's behavior on a PDP-10. One way of dealing with this difficulty is to invent a representative computer and then compare the performances of competing algorithms on that machine. Knuth [1968] has described one such machine which he named the MIX computer; it has much in common with most existing machines without many idiosyncracies of its own. If algorithm A is faster than algorithm B when implemented on MIX, then it is very likely to be faster on most real machines, too.

Another solution to the model of computation problem is not to analyze the implementation of the algorithm on any particular machine at all, but to count only the number of times some *critical operation* is performed. For the analysis of the FFT and matrix multiplication we chose to count the number of arithmetic operations. We know that the FFT uses exactly $n \log_2 n$ multiplications; to estimate its running time for a given implementation we can look up the execution speeds of the instructions around the multiplication instruction, sum those, and then multiply by $n \log_2 n$ to get an estimate for the running time. It is usually easy to determine the running time of a particular program if we know the number of times the critical operation is to be performed[10]. Once we have chosen a critical operation to count it is very easy to specify a model of computation. To count arithmetic operations we usually employ the "straight-line program" model in which an algorithm for a particular value of the problem size (n) is represented by a sequence of statements of the form

$$X_i \leftarrow X_j \text{ OP } X_k$$

where OP is add, subtract, multiply, or divide. If the sequence for a particular value of n is m instructions long then we say that the execution time of our program is $T(n) = m$. If our critical operation were comparison, then we would probably choose the "decision tree" model. These and other models are described by Aho, Hopcroft, and Ullman [1974, chapter 1].

The above models allow us to analyze algorithms for their suitability as "in-core" programs on single-processor machines. If a program has very little main memory available and must store most of its data on tape, then some tape-oriented model such as the "Turing machine" is the most accurate model of the computation. If a program is to be run on a multiprocessor machine then one's model must express this fact; the particular model employed will vary with the multiprocessor architecture[11]. Many other models of computation have been proposed to describe diverse computing devices. The two important things in choosing a model are that it be *realistic*, so the results will apply to the situation it purports to model, and that it be mathematically *tractable*, so we can derive those results.

Exact or Approximate Analysis

Once we have chosen a model of computation we can analyze the performance of an algorithm by counting the amount of resources (time or space) it uses as a function of n, the problem size. How accurately should we do that counting? We could be very precise, calculating the answer exactly, or we might settle for an approximate answer. There are levels of approximation, all the way from the first two terms of the answer to rough upper and lower bounds. It is certainly desirable to get the exact answer, but this is sometimes very difficult. The first one or two terms of the cost function are adequate for most purposes, and in many cases only the asymptotic growth rate of the functions is needed. We saw in the subset testing problem an example in which the run time for one program for a task was 138 hours while another program took just 4 seconds. Even if our analysis missed constant factors of ten, that could not affect our choice for large problems.

---

10 Though we must be careful not to ignore certain "bookkeeping" operations that become critical in implementations.

11 The interested reader is referred to Kung [1976] for a discussion of some of these issues from an algorithmic viewpoint.

We often use the "big-oh" notation to describe the complexity of a problem. No matter what the relative constants are, an $O(n \log_2 n)$ algorithm will be faster than an $O(n^2)$ algorithm for large enough n. As larger and larger problems are being solved by computer we are more and more frequently in the domain of "large enough n". Asymptotically fast algorithms also have another advantage. If we get a new machine one hundred times faster than our current, using an $O(n \log_2 n)$ algorithm will allow us to solve a problem about one hundred times larger in the same period of time. Using an $O(n^2)$ algorithm we will only be able to increase the problem size by a factor of ten. Thus the asymptotic growth rate of a function alone is usually enough to tell us how much an increase in problem size will cost.

Average or Worst-Case Analysis

Many algorithms perform a sequence of operations inependent of their input data; the FFT and matrix multiplication algorithms of Section 2 are both data-independent. The analysis of a data-independent algorithm is straightforward--we count the number of operations used. The operation of other algorithms (such as the sorting and substring algorithms) are dependent on their input data; one algorithm can have very different running times for two inputs of the same size. How do we describe the running time of such an algorithm? Pessimists would like to know the worst-case of the running time over all inputs and realists would like to know the average running time. (We are rarely concerned with the best-case running time, for there are very few optimists involved with computing.)

Most of the mathematical analysis of algorithms has been done for the worst case. Even in data-dependent algorithms it usually easy to identify the worst possible occurrence, and then analyze that as in a data-independent algorithm. In certain applications (Air Traffic Control is often cited) it is very important to have an algorithm with which we are never surprised by a very slow case. For most applications, however, we are more interested in what will usually happen; expected-time analysis provides us with this information. Relatively little work has been done on expected-time analysis. The two major stumbling blocks appear to be in the choice of a realistic and tractable probability model of the inputs and the instrinsic difficulty of dealing with expectations instead of single cases. It would be very desirable to have a single algorithm that is very efficient in both expected and worst-case performances. A family of such algorithms was recently described by Bentley and Shamos [1978]; their expected analysis made very weak probabilistic assumptions.

Upper and Lower Bounds

The first sorting algorithm we mentioned in Section 2.1 required $O(n^2)$ comparisons in the worst case; we then investigated Mergesort, which never uses more than $O(n \log_2 n)$ comparisons. Should we continue our search, hoping to find an algorithm that uses perhaps only $O(n)$ comparisons? The answer to this question is no, for it can be shown that *every sorting algorithm must take at least $O(n \log_2 n)$ comparisons in the worst case.* The proof of this theorem uses the "decision tree" model of computation and is described nicely by Aho, Hopcroft, and Ullman [1974]. The Mergesort algorithm gave us an *upper bound* of $O(n \log_2 n)$ on the complexity of sorting; this theorem gives us a *lower bound.* Since the two have the same growth rate, we can say that Mergesort is *optimal* to within a constant factor, under the decision tree model of computation. Notice that we have now made the important jump from speaking of the complexity of an algorithm to speaking of the complexity of a problem.

Lower bounds are usually much more difficult to obtain than upper bounds. To find an upper bound on a problem one can give a particular algorithm and then analyze it. For a lower bound, however, one must show that *in the set of all algorithms for solving the problem,* there are none which are more efficient than the lower bound. There are some trivial lower bounds which can be achieved easily: most algorithms must examine all their inputs so we usually have an easy lower bound of n (or n^2 for matrix multiplication). The number of nontrivial lower bounds achieved to date is very small.

In proving lower bounds it is important to be very precise about the model of computation. In Section 2.1 we gave three algorithms that can be used for testing set equality[12]: brute force, sorting, and hashing. The importance of computational model becomes clear when we learn that each of those algorithms can be proved optimal under different computational models! The $O(n^2)$ performance of brute force is optimal if only equal/not-equal comparisons can be made between elements of the two sets. If the model of computation iniudes only less-than/not-less-than comparisons then the $O(n \log_2 n)$ comparisons of sorting are optimal. If the model is a random access computer (such as MIX), then the average-case linear performance of hashing is provably best.

Exact and Approximation Algorithms

There are many problems for which the best-known algorithms are quite slow, requiring (say) $O(2^n)$ time. For a very few of these problems people have actually proved lower bounds. Others belong to a fascinating class called the NP-complete problems which are either all solvable in polynomial time or all of exponential complexity--unfortunately nobody knows which (but most of the money is on exponential). Examples of NP-complete problems include the Travelling Salesman Problem (finding a minimal-length tour through a set of cities) and the Knapsack Problem; literally hundreds of problems are known to be NP-complete. There are other problems which have not been proved to be hard, yet no one has been able to design fast algorithms for them. When we have a problem which we do not know how to solve efficiently, what can we do?

The answer is amazingly simple: don't solve it. Solve a related problem instead. Instead of designing an algorithm to produce the exact answer, one can build an algorithm that will produce an approximation to the exact answer. So instead of finding a minimal tour for the Travelling Salesman, we might provide him with a tour which we know to be no more than fifty percent longer than the true minimum. Or if someone asks us to determine if a number is prime or composite, instead of providing the true answer we might respond "I don't know, but I'm 99.999999 percent sure that it's prime." Examples abound in

12 We gave them originally for subset testing, but recall that two sets are equal if and only if each is a subset of the other.

which the best known exact algorithms for a problem require exponential time, but approximate solutions can be found very quickly. Garey and Johnson [1976] examine these issues.

Summary

These problem dimensions are the categories in which algorithm designers think. When someone brings a problem to an algorithm designer, the algorithm designer's first task is to understand the abstract problem. His second task is to understand what kind of solution the person wants, and he uses these dimensions to describe the desired solution. Using the vocabulary of this section it is easy to describe concepts such as a "fast expected time and low worst-case storage approximation algorithm for task X which is to be run on a multi-processor machine". There are certainly other infrequently used dimensions which we have not covered (such as code complexity--how long is the shortest program to solve this problem?) but these dimensions are adequate to describe most algorithmic results.

3.2 Problem Areas--A Macroscopic View

In Section 3.1 we developed a vocabulary which we can use to describe a particular algorithmic problem at a very precise level of detail. In this section we will change our perspective and examine large classes of problems, using the terminology of the last section. We will describe each area by a brief summary and one or two illustrative problems.

Ordered Sets

There are many problems on sets that depend only on a "less than" relationship being defined between the elements of the set. In many cases the set contains integers or real numbers; in other cases we define a "less than" relation between character strings (JONES is less than SMITH). The problems we examined in Section 2.1 are all problems on ordered sets--these include sorting, searching, merging, and subset testing. The algorithms of that section are appropriate if the elements of the sets to be processed are numbers, character strings, or any other type of "orderable" object. Knuth [1973] provides an excellent introduction to the applications of and algorithms for ordered sets.

The "median problem" is defined for ordered sets: given an n-element set we are to find an element which is less than half the elements and not less than the other half. A naive algorithm would count for each element the number of elements less than it, and then report the median as the element with exactly half the others less than it. This data-independent algorithm makes exactly n^2 comparisons. An $O(n \log_2 n)$ algorithm is given by sorting the elements and then reporting the middle of the sorted list. A median algorithm with linear expected time was first described by C. A. R. Hoare in 1962. For over ten years it was not known if there was an algorithm that had linear worst-case time; one was finally given by Blum *et al.* [1973]. Much additional work has been done on this problem, exploring such facets as minimal storage, detailed analysis of worst-case and expected running times (both upper and lower bounds), and approximation algorithms.

Algebraic and Numeric Problems

Many aspects of algebraic and numeric problems have a discrete flavor, and discrete algorithm design can play a significant role in such problems. Matrix multiplication is perhaps the clearest example of such a problem; the fast algorithm can be described (and appreciated) without reference to any of its numeric properties. The FFT can also be viewed "non-numerically". Another example of a numeric problem that can assume a purely discrete character is the manipulation of sparse matrices (matrices in which almost all elements are zero); we return to this problem in our discussion of graph problems. Borodin and Munro [1975] give many applications of the principles of discrete algorithm design to numeric problems such as polynomial manipulation, extended precision arithmetic, and multiprocessor implementations of numeric problems.

Graphs

Graphs are used to represent many different kinds of relations, from the interconnections of an airline system to the configuration of a computer system. Tarjan's [1977] survey discusses many computational problems on graphs. One important problem calls for determining if the nodes of a given graph can be imbedded in the plane without any edges crossing. The first algorithms for testing planarity ran in $O(n^3)$ time on n-node graphs; after much effort on the part of many researchers had been spent on the problem, Hopcroft and Tarjan finally gave a linear-time planarity algorithm in 1974. Another graph problem is to construct the minimal spanning tree of a weighted graph, which is a set of edges connecting all nodes at minimal cost. A wide variety of algorithms have been proposed and analyzed for this problem; some are superior for very dense graphs, others for relatively sparse graphs, and yet others for graphs which are planar. Efficient graph algorithms have been given for problems such as the flow analysis of computer programs and finding maximal flows in networks. A sparse matrix is usually represented by a graph; the algorithms for manipulating matrices are then graph algorithms.

Geometry

Shamos' [1975] paper is an outstanding introduction to the field of *Computational Geometry*, which is concerned with developing optimal algorithms for geometric problems. Many applications are of a geometric nature (such as architects planning a building) and other problems can be viewed geometrically (such as looking at a set of multivariate observations as points in a multidimensional space). Shamos has described an important structure called the Voronoi diagram which allows many geometric problems dealing with n points in the plane to be solved in $O(n \log_2 n)$ time. Among these problems are determining the nearest neighbor of every point and constructing the minimal spanning tree of the point sets (both of these are important tasks of many data analysis procedures, and previously required quadratic time). Many other important problems have been solved after being cast in a geometric framework. One result obtained by this effort is that the simplex method of linear programming is not optimal for two and three variable programs with n constraints. The simplex method has worst-case running times in the two problems of $O(n^2)$ and $O(n^3)$, respectively; an $O(n \log_2 n)$ method has been given and proved optimal for both the two and three variable case.

Summary

In this section I have tried to give the reader some feeling for the scope of the results that have been achieved to date in the field of algorithm design. The results in many areas must go unmentioned; these include algorithms for compilers, operations research problems, data base management, statistics, and problems on character strings. The results described in this section have led to both fast algorithms for solving real problems and to a new, algorithmic understanding of the various fields.

3.3 Fundamental Structures

Wandering through a computer room one can not help but be impressed by the complexity of a large-scale computing system, and the novice might find it hard to believe that a human mind could design anything so complicated. The novice is not too far from the truth, yet many undergraduates are able to understand the basics of the organization of a computer after only one or two semesters. They are able to comprehend the complexity not by sheer force of concentration, but rather by understanding the "building blocks" of which computers are made. A similar experience awaits the novice algorithm designer. The algorithms mentioned in Section 3.2 deal with many problem areas, but are rather simple to comprehend once one understands the "building blocks" of algorithm design. In this section we will describe three important classes of these fundamental structures.

Data Structures

Algorithms deal with data, and data structures are the tools the algorithm designer uses to hold his data. In Section 2 we saw simple data structures such as arrays and matrices, and a fairly complex data structure, the hash table. There are many more exotic types of data structures, such as linked lists, stacks, queues, priority queues, and trees, to name a few. Each of these provides an appropriate way to structure data for a particular task. Tarjan [1977] gives a brief description of many of these structures; a detailed description of a large number of interesting structures is provided by Knuth [1968].

Algorithmic Techniques

Structured programming demands that a programmer express a complicated sequence of commands as a series of refinements by which the program can be understood at different levels. In each of these refinements a basic, well understood method is applied to a well defined problem. Good programmers used this technique long before it was vocalized; good algorithm designers use a similar strategy even though they infrequently discuss it. The constructs available to the algorithm designer are similar to those in structured programming languages, though somewhat more powerful. We will describe some of these constructs very briefly; more detail can be found in Tarjan [1977] and Aho, Hopcroft, and Ullman [1974, ch. 2].

We have already seen many common algorithmic techniques in Section 2. Most of the algorithms we described used *iteration* in one form or another--this strategy says "do x over and over until the task is accomplished". Iteration is present in almost all programming languages as do and while loops. A more powerful construct is *recursion*, which gives us a way to express recursive problem solving in programming languages. To define a recursive solution to a problem one says (essentially), "to solve a problem of a certain size, solve the same problem of a smaller size." We used recursion to describe binary search: to binary search a table of size n we binary searched a table of size n/2. A particular application of recursion is usually called *divide-and-conquer*, and says, "to solve a problem of size n, 1) divide it into subproblems each of size only a fraction of n, 2) solve those subproblems recursively, and 3) combine the subsolutions to yield a solution to the original problem." Mergesort is a textbook example of divide-and-conquer: to sort a list of n elements we 1) break the list into two sublists each of n/2 elements, 2) sort those recursively, and 3) merge those together. The Fast Fourier Transform and the $O(n^{2.81})$ matrix multiplication algorithms are other application of the divide-and-conquer technique. Once one understands the fundamental principles of divide-and-conquer algorithms, each of these instances becomes rather easy to grasp.

Many other algorithmic techniques have been identified and studied. *Dynamic programming* is a technique from operations research that has found many applications in algorithm design. Search strategies such as *breadth-first search* and *depth-first search* have been used to yield efficient graph algorithms. *Transformation* allows us to turn an instance of one problem into another; we saw in Section 2.4 that there are many transformations to turn almost totally unrelated problems into instances of matrix multiplication. Perhaps the single most important algorithmic technique is to use optimal tools to solve the subproblems we create for ourselves in designing a new algorithm. To do so an algorithm designer must keep abreast of the current results in his field.

Proof Techniques

Once an algorithm designer has given an algorithm and "knows in his heart" that it has certain properties, he must prove that it does. (Perhaps it is this step which separates theorist from practitioner.) His first task is to prove that his algorithm indeed computes what it purports to; he will use many of the tools of program verification in this step. Next he must analyze the resource requirements of his algorithm, during which he will use many different mathematical tools. Finally he can prove his algorithm optimal by giving a lower bound proof. The different methods of analysis used in these various steps are discussed by Weide [1977].

4. CURRENT DIRECTIONS

The field of algorithm design has experienced a meteoric rise in the past decade. Essentially unknown as a field ten years ago, it is now one of the most active areas in theoretical computer science and has seen widespread use in applications. Although the field has come a long way, it has much further to go. In this section we will examine some of the directions in which the field is currently moving.

One constant direction of the field has been from "toy" problems to "real" problems. This involves many detailed analyses and expected-resource analyses, for in applications we are often seriously concerned about twenty percent differences in what will usually happen. Along with these efforts much has been done recently on approximation algorithms, for many applications do not require exact answers. On the more theoretical side, the outstanding question is the complexity of the NP-complete problems--are they exponential? Another important theoretical problem is the search for some underlying theory of algorithm design. Though many individual results have been achieved to date, we still have no theoretical explanation for what makes a class of problems easy or hard. Tarjan has mentioned the need for a "calculus of data structures"--a set of rules that will allow us to develop the (provably) best possible structure for a given situation.

An important outgrowth of this work will be the development of "Algorithmic Engineering". This field will supply the programmer with tools similar to those Electrical Engineering gives the circuit designer. Before Algorithm Design turns into Algorithmic Engineering we will need to develop many more particular results and give a theoretical basis for the field. We will know that the field has become an engineering discipline as soon as theoretical computer scientists start to sneer that designing algorithms is no longer *bona fide* research because "It's such a well understood process."

People who work in different applications areas have a serious responsibility in guiding the development of this field. Theoreticians have a strong tendency to work on the problems they can solve at the expense of ignoring the problems important in applications. To help the field from falling into this trap practitioners must tell algorithm designers what their problems are, and where their CPU cycles are going. The author personally would favor the institution of a system whereby a practitioner could complain of high computer costs only after describing his computational problems to at least three people who specialize in algorithm design!

5. CONCLUSIONS

In this paper we have seen the field of algorithm design from a number of different viewpoints. In Section 2 we investigated particular computational problems and their algorithmic solutions. We saw interesting techniques used to solve the problems, learned of many counter-intuitive results, and glimpsed some of the practical benefits of algorithm design. We turned from "war stories" to a systematic view of the field in Section 3. In Section 3.1 we developed a set of terms which can be used to define a computational problem, in Section 3.2 we used those terms to sketch some of the results achieved in the field to date, and in Section 3.3 we mentioned some of the tools used to achieve the results. Having looked at what has already been done in Sections 2 and 3, we turned to the dangerous task of prophecy in Section 4.

In summary I would like to describe what algorithm design has to offer to various individuals. The mathematician and theoretical computer scientist can view the field as a rich source of problems that need precise mathematical treatment; these problems are mathematically fascinating and require the use of some of the most powerful tools of discrete mathematics. The applications programmer with little interest in beautiful theorems can also benefit from this work, for the proper application of its products can occasionally be very rewarding financially. Finally, I feel that anyone involved with computing, regardless of his position on the practical-to-theoretical continuum, should be at least somewhat familiar with this field. The study of algorithms is the study of computing, and through it we gain a fundamental understanding of what computers are all about.

ACKNOWLEDGMENTS

Helpful conversations with Bill Davis, Jay Kadane, Phil Lehman, and Bruce Weide are gratefully acknowledged. I would also like to take this opportunity to thank the "Computer Gang" at Long Beach City College, especially Dave Beers and Rick Lemons, for starting me thinking about how to write good programs.

REFERENCES

Aho, A. V., J. E. Hopcroft, and J. D. Ullman [1974] *The design and analysis of computer algorithms*, Addison-Wesley, Reading, Mass.

Bentley, J. L. and M. I. Shamos [1978]. Divide and conquer for linear expected time, to appear in *Information Processing Letters.*

Blum, M., R. Floyd, V. Pratt, R. Rivest, and R. Tarjan [1973]. "Time bounds for selection," *Journal of Computer and Systems Sciences 7*, 4 (August 1973), 448-461.

Borodin, A. and I. Munro [1975]. *The computational complexity of algebraic and numeric problems*, American Elsevier, N. Y.

Boyer, R. S. and J S. Moore [1977]. "A fast string searching algorithm," *Communications of the ACM 20*, 10, (October 1977), 762-772.

Bullinger, D. R. [1975]. *Time series: Data analysis and theory*, Holt, Rinehart, and Winston, N. Y.

Garey, M. R. and D. S. Johnson [1976]. "Approximation algorithms for combinatorial problems; and annotated bibliography," in *Algorithms and complexity: New directions and recent results*, J. F. Traub, [Ed.], Academic Press, N. Y., pp. 41-52.

Hopcroft, J. E. [1974]. "Complexity of computer computations," in *Proceedings IFIP Congress 74*, vol. 3, North-Holland Publishing Company, Amsterdam, The Netherlands, pp. 620-626.

Karp, R. M. [1972]. "Reducibility among combinatorial problems," in *Complexity of computer computations*, R. E. Miller and J. W. Thatcher, [Eds.], Plenum Press, N. Y., pp. 85-103.

Knuth, D. E. [1968]. *The art of computer programming, vol. 1: Fundamental algorithms*, Addison-Wesley, Reading, Mass.

Knuth, D. E. [1969]. *The art of computer programming, vol. 2: Seminumerical algorithms*, Addison-Wesley, Reading, Mass.

Knuth, D. E. [1971]. "Mathematical analysis of algorithms," in *Proceedings IFIP Congress 71*, vol. 1, North-Holland Publishing Company, Amsterdam, The Netherlands, pp. 135-143.

Knuth, D. E. [1973]. *The art of computer programming, vol. 3: Sorting and searching*, Addison-Wesley, Reading, Mass.

Knuth, D. E. [1977]. "Algorithms", *Scientific American 236*, 4, (April 1977), 63-80.

Knuth, D. E., J. H. Morris, and V. R. Pratt [1977]. "Fast pattern matching in strings," *SIAM Journal of Computing 6*, 2 (June 1977), 323-350.

Kung, H. T. [1976]. "Synchronized and asynchronous parallel algorithms for multiprocessors," in *Algorithms and complexity: New directions and recent results*, J. F. Traub, [Ed.], Academic Press, N. Y., pp. 153-200.

Rabin, M. O. [1976]. "Probabilistic Algorithms," in *Algorithms and complexity: New directions and recent results*, J. F. Traub, [Ed.], Academic Press, N. Y., pp. 21-39.

Reingold, E. M. [1972]. "Establishing lower bounds on algorithms: A survey," in *AFIPS 1972 Spring Joint Computer Conference*, vol. 40, AFIPS Press, Montvale, N. J., pp. 471-481.

Sedgewick, R. [1975] *Quicksort*, PhD Thesis, Stanford University, Stanford, Ca.; for a summary see R. Sedgewick, "The analysis of Quicksort programs," *Acta Informatica 7*, 4 (1977), 327-355.

Shamos, M. I. [1975]. "Geometric Complexity," in *Proceedings of the Seventh ACM Symposium on the Theory of Computing*, ACM, N. Y., pp. 224-233.

Strassen, V. [1969]. "Gaussian elimination is not optimal," *Numerische Mathematik 13*, 345-346.

Tarjan, R. E. [1977]. *Complexity of Combinatorial Algorithms*, Stanford Computer Science Department Report STAN-CS-77-609. To appear in *SIAM Review*.

Traub, J. F., [Ed.] [1976]. *Algorithms and complexity: New directions and recent results*, Academic Press, N. Y.

Valiant, L. G. [1975]. "General context-free recognition in less than cubic time," *Journal of Computer and Systems Sciences 10*, 2 (April 1975), 308-315.

Weide, B. [1977]. "A survey of analysis techniques for discrete algorithms," *Computing Surveys 9*, 4, (December 1977), pp. 291-313.

To .....

Time ..... Date .....

While You Were OUT

M .....

of .....

Phone No. .....

☐ Telephoned ☐ Please call back

☐ Called to see you ☐ Will call again

☐ Left the following message:—

.....

.....

.....

.....

Operator

311

Agreda
Kendall
Thiessen
Eilbert
(2) Warren
Erdem
McV
(2) Stines
Langley ?

ALGORITHMS AND DATA STRUCTURES FOR RANGE QUERIES

Jon Louis Bentley
Departments of Computer Science and Mathematics
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Jerome H. Friedman
Computation Research Group
Stanford Linear Accelerator Center
Stanford, California 94305

ABSTRACT

Problems of data structuring and file organization are discussed when records are to be retrieved associatively on the basis of a range of values for several of their attributes. A variety of methods are described and compared for their suitability in various applications.

1. Introduction

A file is a collection of records, each containing several attributes. A query asks for all records satisfying certain characteristics. If we seek all records for which the attribute values are each within specified ranges, this is called an orthogonal range query. The process of retrieving the appropriate records is called range searching. Conceptually the problem of range searching can be cast in geometric terms. One can regard the record attributes as coordinates and the k values for each record as representing a point in a k-dimensional coordinate space. The respective query ranges can be represented as a k-dimensional hyperrectangle in this space. The problem of range searching is then to find those points lying inside this hyperrectangle.

Range searching arises in many applications. A university administrator may wish to know those students whose age is between 21 and 24 years and whose grade point average is greater than 3.5. From a file of all U.S. cities a list of all those for which the latitude is between 37° and 41° and longitude between 102° and 109° (defining the state of Colorado) may be sought. In data analysis it is often useful to do separate analyses on sets of data lying in different regions of the data measurement space and then compare (or contrast) the respective results. In statistics range searching can be employed to determine the empirical probability content of a hyperrectangle, to determine empirical cumulative distributions, and to perform density estimation (Loftsgaarden and Quesenberry, 1965).

Files to be searched may reside on a variety of physical media. If it is not too large the file can be accommodated in the random access memory of a computer. Larger files may require random access disk drives for their storage. Exceptionally large files may be accommodated only by a series of magnetic tapes. A similar situation arises for data stored on cassette drives of a microprocessor configuration.

There are several problems closely related to range searching for which there has been considerable research. Future researchers might apply the methods used for solving these problems to the problem of range searching. Bentley (1975a) discusses the problem of finding all points within a fixed radius of a given point. Yuval (1975) and Bentley, Stanat, and Williams (1977) discuss this problem for the special case of the L_∞ metric. Friedman, Bentley, and Finkel (1977) discuss the problem of finding the k-nearest neighbors of a point in a file of N points. Bentley (1976) discusses the problem of finding the nearest neighbor to all N points in the file. Domination problems are also closely related to range searching. A point is said to dominate another if all of its coordinates are larger. King, Luccio, and Preparata (1975) discuss the determination of whether a given point is dominated by any other point. Bentley and Shamos (1977) discuss the calculation of how many points a given point dominates, which is the empirical cumulative distribution evaluated at the point.

2. Logical Structures

In this section we discuss the various methods for range searching in terms of their logical structures; that is the logical structure of the data at the level of adjacency and "pointers" without regard to implementation. We devote Section 3 below to the problem of how one implements these logical structures on specific storage media.

A search method is specified by a data structure for storing the data, and algorithms for building the structure (which we call preprocessing), searching the structure, and possibly various utility operations such as insertion and deletion. One analyses a search structure (algorithm) on the basis of 1) the cost of preprocessing N-points in k-space, $P_S(N, k)$; 2) the storage required, $S_S(N, k)$; and 3) the search time or query cost, $Q_S(N, k)$. These costs can be analyzed in terms of their average or their worst case cost. We will usually speak of the worst case cost.

2.1 Brute Force

The simplest approach to range searching is to store each of the N points in a sequential list. As each query arrives all members of the list are scanned and all records that satisfy the query are enumerated. If the queries do not have to be handled immediately they can be batched so that many queries can be processed with one sequential pass through the file.

It is easy to see that the brute force structure, B, possesses the following properties:

$$P_B(N, k) = O(Nk)$$
$$S_B(N, k) = O(Nk)$$
$$Q_B(N, k) = O(Nk).$$

Brute force searching has the advantage of being trivial to implement on any storage medium. It is competitive with the more sophisticated methods described below when the file is small and the number of attributes is large, or when a large fraction of the records in the file satisfy the query (or queries, if they are batched).

2.2 Projection

The projection technique is referred to as inverted lists by Knuth (1973). This technique was applied by Friedman, Baskett, and Shustek (1976) in their solution of the nearest neighbor problem. Projection involves keeping, for each attribute, a sequence of the records in the file sorted by that attribute. One can view this geometrically as a projection of the points on each dimension. The k lists representing the project-

can be obtained by using a standard sorting algorithm. After preprocessing, a range query can be answered by the following search procedure: chose one of the attributes, say the i-th. Look up the two positions in the i-th sequence (using a binary search) of the extreme values defining the range on the i-th attribute of the query. All records satisfying the query will be in the list between these two positions just found. This smaller list is then searched by brute force.

One can apply the projection technique with only one sorted list. If the distribution of values of the various attributes are more or less uniform over similar ranges and the query ranges of each attribute are similar, then one list is sufficient. If not, then it can pay to keep sorted sequences on all k attributes. The positions of the corresponding query range extremes are found in each of the k lists. The list for which the difference in positions is smallest is searched between the two positions.

Analysis of the projection technique, P, for near neighbor searching is reported in Friedman, Baskett, and Shustek (1976). Most of this analysis directly carries over to the problem of range searching. It is clear that

$$P_p(N, k) = O(kN\log N),$$

and

$$S_P(N, k) = O(kN).$$

For searches that find a small number of records (and are therefore similar to near neighbor searches) one has

$$Q_p(N, k) = O(N^{1-1/k}).$$

The projection technique is most effective when the number of records satisfying each query is usually close to zero.

2.3 Cells

"There are two ways they can search [for the murder weapon]: from the body outward in a spiral, or divide the room up into squares -- that's the grid method".

From CBS series Kojak
"Death Is Not a Passing Grade".

Cartographers as well as detectives use the grid (or cell) method. Street maps of many metropolitan areas are printed in the form of books. The first page of the book shows the entire area and the remaining pages are detailed maps of (say) one-mile square regions. To find, for example, all schools in a specified rectangle one would look in the first page to find which squares overlap the rectangle and then check only on those pages to find the schools. This approach can be mechanized immediately. A square of the map corresponds to a cell in k-space, and the points of the file within the cell are stored as a linked list. The first page of the map book corresponds to a directory which allows us to take a hyper-rectangle and look up the set of cells.

Knuth (1973) has discussed this scheme for the two-dimensional case. Levinthal (1966) used a cell technique in three dimensional Euclidean space for determining all atoms within five angstroms of every atom in a protein molecule -- he referred to this as "cubing". Yuval (1975) and Rabin (1976) apply an overlapping cell structure to the near neighbor problem.

The directory can be implemented in two ways. If the points are (say) uniformally distributed on $[0, 10]^2$ and we have chosen 1 x 1 cells, we can use a two-dimensional array as the directory. In DIRECT (i, j) we would keep a pointer to a list of all points in the cell $[i, i + 1] \times [j, j + 1]$. If we then wanted to find all points in $[5.2, 6.3] \times [1.2, 3.4]$

then we would only have to examine cells (5, 1), (5, 2), (5, 3), (6, 1), (6, 2), (6, 3). The multidimensional array works very well when the points are known a priori to be in a rectangle. When this is not known to be the case, one would probably use a search method, such as hashing, for the directory. In this method we name each cell as before, so cell (i, j) is a pointer to the points in (i, i + 1) x (j, j x 1). Instead of storing all cells, however, we store only cells which contain points of the file. To process a query we "decode" the rectangle into a set of cell id's, look up those id's, and check the points in the occupied cells for inclusion in the rectangle. The storage required for the cell technique is the storage for the directory plus locations for the linked list representing points in cells; the size of the directory is usually much smaller than N.

Basic parameters of the cell technique are the size and shape of each cell. In analyzing a search there are two costs to count: cell accesses (the number directory look-ups) and inclusion tests (testing whether a point satisfies the range query). If the cells are extremely large, there will be few cell accesses and many inclusion tests. If the cell size is very small, on the other hand, there will be very many cell accesses and very few inclusion tests. Clearly either extreme is bad.

The best cell size and shape depends upon the size and shape of the query hyperrectangle. Bentley, Stanat, and Williams (1978) show that if the query hyperrectangles have constant size and shape so that only their location (in the coordinate space) is unspecified, then for a single grid a nearly optimum size and shape for the cells are the same as that for the query hyperrectangle. For this case the number of cells accessed is 2^k and the expected search time is proportional to 2^k times the number of points in the range. In most applications the queries will vary in their size and shape as well as their location, so that there is little information available as to how to make a choice of cell size and shape.

2.4 k-d Trees

This data structure was introduced by Bentley (1975b). Friedman, Bentley and Finkel (1977) introduced adaptive k-d trees and showed that this structure is very effective for nearest neighbor searching. The application of k-d trees has the effect of dividing the k-space into a collection of irregular hyperrectangles each with the property that they are approximately cubical and all contain nearly the same number of points. This overcomes the problem of empty cells which severely limits the performance of searching with regular grids. The cell pattern induced by k-d trees adapts to the distribution of the points in the k-space.

The k-d tree is a generalization of the simple binary tree used for sorting and searching. The k-d tree is a binary tree in which each node represents both a subcollection of the points in the space and a partitioning of that subcollection. The root of the tree represents the entire collection. Each nonterminal node has two successors. These successor nodes represent the two subcollections defined by the partitioning. The terminal nodes represent mutually exclusive small subsets of the points, which collectively form a partition of k space. These terminal subsets are called buckets.

In the case of one-dimensional searching, a point is represented by its value in a single dimension and a partition is defined by some value of that dimension. All records in a subcollection with key values less than or equal to the partition value belong to the left successor node, while those with a larger

value belong to the right successor. The dimension thus becomes a discriminator for assigning records to the two subcollections. In k-space, a point is represented by its k dimensions. Any one of these can serve as the discriminator for partitioning the subcollection represented by a particular node in the tree; that is, the discriminator can range from 1 to k.

The prescription for constructing an adaptive k-d tree is to choose for the discriminator that coordinate i for which the spread of attribute values (as measured by any convenient statistic) is maximum for the subcollection represented by the node. The partitioning value is chosen to be the median value of the ith attribute. This prescription is then applied recursively to the two subcollections represented by the two sons of the node just partitioned.

The partitioning is stopped, creating a terminal node or bucket when the cardinality of the subcollection is less than a prespecified maximum. This maximum is a parameter of the procedure. Friedman, Bentley, and Finkel (1977) found empirically that values ranging from 8 to 16 worked best for nearest neighbor searching.

Associated with each node in a k-d tree is a set of geometric bounds within which the points in that subcollection (represented by the node) must lie. The bounds can be represented by two one-dimensional arrays of k elements, called LOWER and UPPER. For a given subcollection S it must be true for every point x in S and every coordinate i that

$$\text{LOWER}(i) \leq x_i < \text{UPPER}(i).$$

The bounds are updated as the recursive partitioning proceeds. That is, if at a particular node a partition is made on coordinate i at position p, then UPPER (i) ← p for the subcollection represented by the left son of the node, while LOWER (i) ← p for that represented by the right son. The LOWER and UPPER arrays are initialized to minus and plus

Range searching with k-d trees is straightforward. Starting at the root the k-d tree is recursively searched in the following manner. At each node visited a decision is made as to whether it is necessary to search either of its sons. If the bounds (LOWER, UPPER) on any coordinate (attribute) are outside the query range on that coordinate, the subcollection represented by that node are all outside the query range and the node (and all of its descendants) need not be searched. If the bounds on all coordinates are entirely within the query bounds then the entire subcollection represented by the node is within the range and satisfy the range query. In that case the records representing the subcollection are simply enumerated and the node (and all of its descendants) need not be searched. A node need only be visited (searched) if one or more of its coordinate bounds lie partially within the corresponding range of the query. If it is determined that a son must be visited, the above procedure is applied (recursively) to the bounds of each of its sons to see if they must be visited. With this strategy only those file records contained within k-d tree cells that enclose part of range query boundary are explicitly searched. A pseudo-ALGOL procedure for range searching of k-d trees is described in (Bentley, 1975b).

Analysis of k-d trees for range searching has been considered by several researchers. The work required to construct a k-d tree, its storage requirements are:

$$P_K(N, k) = O(N \log N), \quad \text{and}$$
$$S_k(N, k) = \text{a few percent of } N.$$

The search cost depends upon the nature of the query. In the very worst case Lee and Wong (1976) show that

$$Q_k(N, k) \leq O(N^{1 - 1/k}) \quad \text{(worst case)}.$$

If the number of records that satisfy the query is small so that the range query is similar to a near-

est neighbor search one has from Friedman, Bentley and Finkel (1977).

$$Q_k(N, k) = O(\log N) \quad \text{(average case for small answer)}$$

+ points in region.

For the case where a large fraction of the file satisfies the query Bentley and Stanat (1975) show

$$Q_k(N, k) = \text{points in region} + \text{small terms.} \quad \text{(average case for large answer)}$$

The k-d tree structure is most effective in situations where little is known about the nature of the queries or a wide variety of queries are expected. They are also useful if, in addition to range queries, near neighbor or spherical region queries are anticipated.

2.5 Range Trees

In this section we describe the range tree, a structure introduced by Bentley (1977). It achieves the best search time of all the structures we have seen so far, but has relatively high preprocessing and storage costs. For most applications the high storage will be prohibitive, but the range tree is very interesting from a theoretical viewpoint. Since the range tree is defined recursively we will begin our discussion by looking at a one-dimensional structure, and then generalize that structure to higher dimensions.

The simplest structure for one-dimensional range searching is a sorted array. The preprocessing sorts the N elements to be in ascending order by key. To answer a range query we do two binary searches to find the positions of the low and high end of the range in the array. After these two positions have been found we can list all the points in that part of the array as the answer to the range query. For this structure we use linear storage and O(N log N) preprocessing time. The two binary searches will each cost O(log N), and the cost of listing the points found in the region will, of course, be proportional to the number of such points. Letting F be the number of points found in the region, we have

$$P_R(N,1) = O(N \lg N),$$

$$S_R(N,1) = O(N), \text{ and}$$

$$Q_R(N,1) = O(\lg N + F).$$

We will now build a two-dimensional range tree, using as a tool the one-dimensional sorted arrays we described above (which we abbreviate SA's). The range tree is similar to the "binary search trees" described by Knuth (1973, Section 6.2) so we will use his terminology in our discussions. The range tree will be a rooted binary tree in which every node has a left son, a right son, a discriminating value (all nodes in the left subtree have a discriminating value less than the node's) and (unlike a regular binary search tree) every node contains an SA (sorted by Y-coordinate) and has as discriminating value the median x-value for all points. The left subtree of the root has an SA containing the N/2 points with x-value less than median sorted by y-coordinate. Likewise the left son of the root represents the N/2 points with x-value greater than the median and has an SA of those points sorted by y-coordinate. This partitioning continues so that i levels away from the root we have 2^i subtrees, each representing $N/2^i$ points contiguous in the x-coordinate, and each containing an SA of the points sorted by y-coordinate. This partitioning continues for a total of (approximately) lg N levels; we handle small point sets (say less than a dozen points) by brute force.

The search algorithm for a range tree is most easily described recursively. Each node in the tree represents a range in the x-dimension. When visiting a node we compare the x-range of the query to the range of the node, and if the node's range is entirely within the query's then we search that structure's SA for all points in the query's

y-range. After this we compare the query's x-range to the node's discriminator value. If the range is entirely below the discriminator we recursively visit the left subtree; if it is above we visit the right; and if the range overlaps the discriminator then we visit both subtrees.

The analysis of the planar tree is somewhat complicated. Since there are lg N levels in the tree and N points are stored on each level, the total storage required is O(N lg N). The preprocessing can be performed in O(N lg N) time if some clever techniques are employed. Analysis shows that at most two SA searches are done on each level of the tree (each of cost approximately lg N) so the total cost for a search is $O(lg^2N)$ plus the time for listing the points in the region. Letting F stand, as before, for the total number of points found in the region we have

$$P_R(N,2) = O(N \lg N),$$
$$S_R(N,2) = O(N \lg N), \text{ and}$$
$$Q_R(N,2) = O(lg^2N + F).$$

If we step back for a moment we can see how we built the structure: we constructed a two-dimensional structure by building a tree of one-dimensional structures. We can perform essentially the same operation to yield a three-dimensional structure: we construct a tree containing two-dimensional structures in the nodes. This processs can be continued to yield a structure for k-dimensions, which will be a tree containing (k-1)-dimensional structures. This results in structure with performances.

$$P_R(N,k) = O(N\ lg^{k-1}N),$$
$$S_R(N,k) = O(N\ lg^{k-1}N), \text{ and}$$
$$Q_R(N,k) = O(lg^kN + F).$$

The range tree structure is very interesting from a theoretical viewpoint. The asymptotic search time is very fast but the amount of storage used is probably prohibitive in practice. So although the application of this structure to practical problems will probably be limited to cases when k = 2 or 3, it does provide an important theoretical benchmark. It also gives us an interesting method that might yield fruit in practice. (Indeed, there are some very interesting relationships between range trees and the k-d trees of Section 2.4).

2.6 Other Structures

In this section we briefly mention several structures that we feel are no longer competitive with those discussed above. We include them for completeness and in the hope that someone might be inspired by one of them to invent techniques superior to those we have discussed.

Knuth (1973) points out that the notion of cells can be applied recursively. That is, when one of the cubes has more than some certain number of points, the cube is further divided into subcubes of yet smaller size. This scheme implies a multidimensional tree with multiway branching. In terms of both the partitioning imposed on the space and the ease of implementation, this idea seems to be dominated by the quad tree (see below), which is in turn dominated by the k-d tree.

Finkel and Bentley (1974) describe a structure called the quad tree. It is a generalization of the binary tree in which every node has 2^k sons. Bentley and Stanat (1975) analyzed the performance of quad trees for fixed radius near neighbor searches in 2-space using the maximum coordinate metric in uniform point sets. Linn (1973) discussed the fact that quad trees (which he called "Search-sort k trees") have advantages over binary trees when used in a synchronized multiprocessor system. This application aside, however, the quad tree seems to be dominated by its historical successor, the k-d tree.

Bentley and Shamos (1977) describe a data structure (the ECDF tree) for finding the empirical cumulative distribution of a point (in k-dimensional space) among a collection of points. If only a count of the number of points in the query hyperrectangle is required and, not a listing of the points, then several ECDF searches can be used to obtain that count. This structure has very desirable worst case performance but requires considerable storage and has average-case behavior worse than k-d trees when applied to range searching.

2.7 Comparison of Methods

Four structures (brute force, projection, cells, and k-d trees) have been presented as providing practical solutions to the range searching problem. For each there are situations for which it is clearly superior and other situations where it performs badly. In this section we try to enumerate various situations and compare the performance of the four methods.

If the file is small and the number of attributes large, if the file is to be searched only a few times, or if the queries can be batched so that nearly all of the records in the file satisfy at least one, then brute force is the method of choice. Otherwise one of the other methods is likely to be more efficient. Projection does best when the query range on only one of the attributes is sufficient to eliminate nearly all of the file records. For this case the low overhead of searching this structure allows it to dominate the others. In situations where several or many of the attributes serve to restrict the range query the projection technique performs badly.

The cell and k-d tree structures are appropriate in those situations where the query restricts several or many of the attributes. If the approximate size and shape of the queries are roughly constant and are known in advance then cells defined by a fixed grid with size and shape common to that of the expected queries is most advantageous. However, for queries with sizes and shapes that differ considerably from the design, performance is poor.

The k-d tree structure is characterized by its robustness to wildly varying queries. The cell design adapts to the distribution of the attribute values of the file records in the k-dimensional coordinate space. The cells all contain very nearly the same number of records; there are no empty cells. In dense regions there are many cells and a fine division of the coordinate space; in sparse regions there is a coarser division with fewer cells. If a wide variety of queries are expected then the k-d tree structure should serve best.

3. Implementations

In Section 2 we discussed the various structures for range searching in a more or less abstract way without regard to implementation. We now turn our attention to how one implements these structures on real computers.

3.1 Internal Memory

If the file is small enough so that it can be contained in the internal memory of the computer then implementations of these structures is straightforward. The brute force structure is implemented as a two dimensional (N x k) array. For projection one has a set of tables of pointers to each record; each table is sorted on a different coordinate.

As discussed in Section 2.3, there are two possible ways to associate records with cells. If the points are uniformly distributed in a more or less rectangular area so that there are few empty cells then the grid can be efficiently represented as a multidimensional array. If there are many empty cells then the k attribute values defining

a cell can be treated collectively as a key and a well known search method such as binary searching or hashing can be employed.

The k-d tree can be implemented as any other binary tree [see Knuth (1973) and Bentley, (1975b)]. It is easy to store for each node a pair of pointers to the records defining the subcollection associated with the node. This facilitates enumeration of the records satisfying the query (if this is the case for all records of the node) without traversing the descendants of the node.

3.2 Disk

Implementing these structures on random access disks is only slightly less straightforward than for central memory. For the most part disk addresses simply replace memory addresses. For brute force one simply performs a sequential scan of the records. With projection the sorted lists contain pointers to the disk address of the corresponding records. The lists for each attribute can themselves be stored on the disk and only one list at a time need reside in central memory. With the cell technique the hash tables contain disk pointers and reside in central memory. Only those cells overlapping the query range need be read into central memory.

Tree structures lend themselves nicely to external searching from random access disks. This is discussed by Knuth (1973, p 472). Figure 1 is a reprint of figure 29 from Knuth. The nodes of the tree are grouped (as shown by the dotted lines) into "pages". The size of each page is chosen as some convenient unit of disk memory (such as a track or sector). While the tree is beingsearched in the usual manner only one page at a time need reside in central memory. If the records satisfying the query represent a small fraction of the file then on the order of

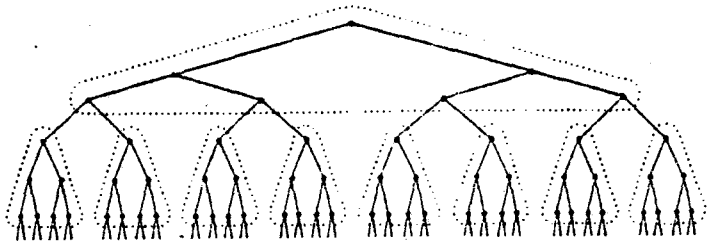


Fig. 1. A large binary search tree can be divided into "pages."

log(N/b) disk accesses are required where b is the page size. Williams et al (1975) have implemented k-d trees for range searching on a random access disk system.

3.3 Tape

By its nature magnetic tape is a sequential storage medium. This makes it ideal for the brute force approach. However, even within this sequential limitation, it is possible to employ to advantage the other range searching methods described above. In order to read a record from a magnetic tape it is necessary to pass over all records from the beginning to it. However, it is not necessary to read all of those records into central memory or even transmit them from the controller to the channel. On most computing systems it is possible to issue instructions to skip one or several blocks without transmitting any data. Although the real time to read a tape is nearly the same whether blocks are skiped or read, the CPU requirement memory interference, and channel activity can be substantially reduced. This is important in a multiprogramming environment.

With the projection method the first k-blocks on the tape are the lists of pointers to the data records sorted on each attribute. This is followed by the data records. Each sorted list is read into central memory in turn and the list of records to be searched is determined as described in Section 2.2. These records are then read sequentially from the tape skipping all records in between them.

The cell method is implemented similarly. Here the directory comprises the first few blocks of the tape with the data following, arranged so that points within each cell comprise one block of data. The cells overlapping the query are determined from the directory and then those cells are read sequentially from the tape skipping unwanted blocks.

The hierarchical nature of k-d trees and range trees allows for a natural implementation on a sequential storage medium such as magnetic tape. The nodes of the tree are stored in the order of a preorder (left son, right son) traversal of the tree. Each node comprises a record. The terminal nodes are the data records. Associated with each node (record) are the geometric bounds (LOWER, UPPER) that delimit the data records that are its descendants in the tree. Also associated with each node is the number, D, of its descendants.

With this arrangement the tree search can proceed directly from the tape. At each node visited (beginning with the root which is the first record on the tape) a determination is made as to whether it is necessary to search either of its sons, as described in Section 2.4. If the bounds associated with a node are outside the range of the query then a command is issued to skip D records on the tape. If the bounds are entirely within the query range then all data records within the next D tape records are in turn presented as satisfying the query. If the bounds are partially within the query we have three cases. The easiest is when both sons are to be visited--continue reading the tape. If only the right son is to be visited, this is also easy--skip the number of blocks occupied by the left subtree. The case of visiting the left subtree only is more complicated; here one stacks the number of records to be skipped when returning to this node, and when control is returned to the node that many blocks are skipped.

With this method the number of blocks read into main memory is equal to the number of nodes visited in the tree search. This technique can be applied to a wide variety of tree searches on tape and the same behavior will be obtained. In particular this method can be used with the range trees of Section 2.5, in that magnetic tape can often accommodate their large storage requirements.

4. Further Work

Our discussion of range searching has in many respects just scratched the surface and there are many avenues open for further research.

All files that we have discussed so far have been static, that is, unchanging. Many applications require dynamic structures, in which insertions and deletions can be made. Dynamic versions of brute force, projection, and cell structures are easily obtained. Dynamic k-d trees are briefly discussed by Bentley (1975b). Considerable work remains to be done in the dynamic analysis of all of these structures.

Considered research also remains in the development of heuristics for aiding these search methods. For example, if in a seven dimensional problem the range queries almost always involve only two of the attributes, then the design of the structure for projection, cell, k-d tree or range trees should only involve these two attributes. Heuristics for detecting these and other similar situations would be very helpful.

5. Conclusion

In this paper we have tried to present a reasonably complete survey of the state of the art for range searching. We have presented a number of results that may be used by those who implement such systems, and we have described a

number of methods that can be of use to data base designers.

REFERENCES

Bentley, J. L. [1975a]. A survey of techniques for fixed radius near neighbor searching, Stanford Linear Accelerator Center Report SLAC-186, August 1975, 33 pp.

Bentley, J. L. [1975a]. "Multidimensional binary search trees used for associative searching", Communications of the ACM, vol. 18, no. 9, September 1975, pp. 509-517.

Bentley, J. L. and D. F. Stanat [1975]. "Analysis of range searches in quad trees", Information Processing Letters, vol. 3, no. 6, July 1975, pp. 170-173.

Bentley, J. L. and W. A. Burkhard [1976]. "Heuristics for partial match retrieval data base design", Information Processing Letters, vol. 4, no. 5, February 1976, pp. 132-135.

Bentley, J. L. and M. I. Shamos [1976]. "Divide and conquer in multidimensional space", Proceedings of the Eighth Symposium on the Theory of Computing, ACM, May 1976, pp. 220-230.

Bentley, J. L. [1976]. Divide and conquer algorithms for closest-point problems in multidimensional space. Ph.D. Thesis, University of North Carolina, Chapel Hill, North Carolina.

Bentley, J. L. [1977a]. Decomposable searching problems, extended abstract, Carnegie-Mellon University Computer Science Department.

Bentley, J. L. and M. I. Shamos [1977b]. "A problem in multivariate statistics: algorithm, data structure, and applications", Proceedings of the Fifteenth Allerton Conference on Communication, Control and Computing, to appear.

Bentley, J. L., D. F. Stanat, and E. H. Williams, Jr. [1978]. The complexity of near neighbor searching, to appear in Information Processing Letters.

Dobkin, D. and R. J. Lipton [1976]. "Multidimensional searching problems", SIAM Journal of Computing 5, pp. 181-186.

Finkel, R. A. and J. L. Bentley [1974]. "Quad trees--a data structure for retrieval on composite keys", Acta Informatica, vol. 4, no. 1, pp. 1-9.

Friedman, J. H. [1975]. A variable metric decision rule for nonparametric classification, Stanford Linear Accelerator Center Report SLAC-PUB-1573, April 1975, 30 pp.

Friedman, J. H., F. Baskett, and L. J. Shustek [1975]. "An algorithm for finding nearest neighbors". IEEE Transactions on Computers, vol. C-24, no. 10, October 1975, pp. 1000-1006.

Friedman, J. H., J. L. Bentley, and R. A. Finkel [1977]. An algorithm for finding best matches in logarithmic time, ACM TOMS ACM Transactions on Mathematical Software, Vol. 3, No. 3, September 1977, pp. 209-226.

Knuth, D. E. [1973]. Sorting and Searching, The Art of Computer Programming, vol. 3, Addison-Wesley, Reading, Massachusetts.

Knuth, D. E. [1973]. The Art of Computer Programming, vol. 3: Sorting and Searching, Addison-Wesley, Reading, Massachusetts.

Kung, H. T., F. Luccio, and F. P. Preparata [1975]. "On finding the maxima of a set of vectors", Journal of the ACM, vol. 22, no. 4, October 1975, pp. 469-476.

Lee, R. C. T., Y. H. Chin, and S. C. Chang [1975]. Application of principal component analysis to multi-key searching, National Tsing Hua University, Republic of China, 28 pp.

Lee, D. T. and C. K. Wong [1978]. Worst-case analysis for region and partial region searches in multidimensional binary search trees and quad trees, IBM Watson Research Center preprint, 18 pp. To appear in ACTA Informatica.

Loftsgaarden, D. O. and C. P. Quesenberry [1965]. "A nonparametric density function", Annals of Mathematical Statistics, vol. 36, pp 1049-1051.

Yuval, G. [1975]. "Finding near neighbors in K-dimensional space", Information Processing Letters, vol. 3, no. 4, March 1975, pp. 113-114.

Space-Efficient On-Line Selection Algorithms

Bruce W. Weide *

Carnegie-Mellon University

Abstract - Algorithms for the "order statistics problem", which is essentially a statistical problem, and for the "selection problem" from computer science, require linear space. We present on-line algorithms for these problems which are space-efficient (i.e., require sub-linear direct access storage) but which consequently give only approximately correct results. It is shown that the approximations are quite good.

I. INTRODUCTION

Although linear-time algorithms which find the median element of a totally ordered set are known, these algorithms require linear space (see Knuth [1973], Section 5.3.3, Exercise 15). In particular, even if the elements of the set are generated (or can be transferred to primary memory of a computer) sequentially, we must in general have direct access storage for n/2 elements in order to find the median element exactly. Here we consider whether we might not be able to take advantage of the "on-line" appearance of the set of elements, thereby saving primary memory space, if we are willing to settle for an approximate answer to the problem.

If we note the ultimate reason for finding the median element of a set, it is clear that in many applications an approximate answer is sufficient. There are really two separate problems:

(1) Given a set of n real numbers which are observations from some unknown population, estimate the median (or, in general, the $(100\alpha)^{th}$ percentage point) of the population distribution. We will call this the "order statistics problem." Note that it does not require that the estimate be obtained by finding the $(\lfloor \alpha n \rfloor + 1)^{th}$-smallest observation, or for that matter, that the estimate be one of the observations at all.

(2) Given a set of n linearly ordered elements, find the median (or, in general, the $(\lfloor \alpha n \rfloor + 1)^{th}$-smallest) element of the set. We require the answer to be one of the original elements. This is the "selection problem." In this paper we will consider only the "median selection problem," which is somewhat easier than the general case.

We will demonstrate on-line algorithms for both problems which use very little space, without sacrificing time, but in the second case sacrificing accuracy. In particular, the algorithm for the median selection problem will not be guaranteed to find the median exactly, but will (with specified probability) find an element whose rank is very close to (n+1)/2. This

* Supported by an IBM Graduate Fellowship.

problem therefore provides another example of the time-space-accuracy trade-off, and shows that it is possible that any two of the three factors may be traded off independently of the remaining one. The ordinary time-space trade-off and a time-accuracy trade-off demonstrated by most approximation algorithms are the other possibilities.

To summarize the results, we will exhibit algorithms which demonstrate the following theorems:

THEOREM 1 (Order statistics problem): Let the population distribution F(x) be absolutely continuous and strictly increasing whenever $0 < F(x) < 1$; let the density $f(\xi_\alpha) > 0$, where ξ_α is the population $(100\alpha)^{th}$ percentage point; and let F have a finite absolute δ^{th} moment for some $\delta > 0$. Then for any function $r(n) \to \infty$, it is possible to construct an estimate of the population $(100\alpha)^{th}$ percentage point having the following characteristics:

(1) The estimate has the same asymptotic distribution as the $(\lfloor \alpha n \rfloor + 1)^{th}$-smallest observation.

(2) The estimate can be computed on-line in O(n) time and r(n) space.

THEOREM 2 (Median selection problem): For any odd integer $r > 1$, it is possible to produce an estimate of the sample median (i.e., to report as the approximate answer some element of the set) having the following properties, if each possible permutation of the elements is equally likely to be the input stream:

(1) The estimate is unbiased (i.e., the expected relative rank of the element reported is 1/2).

(2) The variance of the relative rank of the element reported is $O(n^{-1} + \log(\pi/2)/\log r)$.

(3) The estimate can be computed on-line in O(n) time and r log n / log r space.

II. THE ORDER STATISTICS PROBLEM

Let $\{X_i\}$ ($i = 1, 2, 3, \ldots, n$) be a random sample from an unknown population with density function f(x) and cumulative distribution function F(x). Let $X_{(k)}$ denote the k^{th}-smallest element of $\{X_i\}$, and $X_{(n,\alpha)}$ the $(\lfloor \alpha n \rfloor + 1)^{th}$-smallest. Let $F(\xi_\alpha) = \alpha$, so that ξ_α is the $(100\alpha)^{th}$ percentage point of F. We assume that F(x) is absolutely continuous and strictly increasing whenever $0 < F(x) < 1$, that $f(\xi_\alpha)$ exists and is positive, and that F has a finite absolute δ^{th} moment for some $\delta > 0$.

Walker [1968] shows that under these conditions $n^{1/2}(X_{(n,\alpha)} - \xi_\alpha)$ converges in distribution to a normal distribution with mean 0 and variance $\alpha(1-\alpha)/f(\xi_\alpha)^2$, which we will write as $n^{1/2}(X_{(n,\alpha)} - \xi_\alpha) \to_d \mathfrak{N}(0, \alpha(1-\alpha)/f(\xi_\alpha)^2)$. This result not only allows us to determine how good an estimate $X_{(n,\alpha)}$ is, but provides the key idea for a space-efficient algorithm which also estimates ξ_α. The intuitive idea for the case $\alpha = 1/2$ is that since the median equals the mean for any symmetric distribution which has a mean (and therefore for the normal distribution), and the sample mean is known to be the minimum-variance unbiased estimator of this parameter for the normal distribution, we should be able to "replace" median-finding by averaging at some point. Averaging can obviously be done using constant space; this fact, combined with the superiority of the sample mean as an estimate of the center of a normal distribution, allows us to prove Theorem 1.

ALGORITHM 1 (Order statistics problem): Let $r(n) \to \infty$ with r(n) and n/r(n) integers. Here, $n = |S|$.

```
procedure estimate(S,α);
begin
    set Q;
    real total := 0;
    for j := 1 until n/r(n) do
        begin
            Q := next r(n) elements of S;
            total := total + ((⌊αr⌋+1)th-
                    smallest element of Q)
        end;
    return(total * r / n)
end
```

This algorithm is implemented in such a way that only r(n) space is used if the elements of S are generated or transferred to primary memory on-line. The time required is easily seen to be O(n). If we let $Z_{n,\alpha}$ be the estimate produced by Algorithm 1, then all that remains to be shown is that $n^{1/2}(Z_{n,\alpha} - \xi_\alpha)$ converges in distribution to $\mathfrak{N}(0, \alpha(1-\alpha)/f(\xi_\alpha)^2)$.

Let $Y_{r,\alpha,j}$ be the $(\lfloor \alpha r \rfloor + 1)^{th}$-smallest element of the j^{th} subset. Then $r^{1/2}(Y_{r,\alpha,j} - \xi_\alpha) \rightarrow_d \mathfrak{N}(0, \alpha(1-\alpha)/f(\xi_\alpha)^2)$; furthermore, the $Y_{r,\alpha,j}$ are independent because the original observations $\{X_i\}$ are independent. Sen [1959] has shown that, under the conditions of the theorem, the $Y_{r,\alpha,j}$ have finite mean and variance which approach those of the limiting normal distribution. This fact allows us to apply the central limit theorem to show that $n^{1/2}(Z_{n,\alpha} - \xi_\alpha) = (n/r)^{1/2}((r/n)\sum_j r^{1/2}(Y_{r,\alpha,j} - \xi_\alpha)) \rightarrow_d \mathfrak{N}(0, \alpha(1-\alpha)/f(\xi_\alpha)^2)$, which proves Theorem 1.

III. THE MEDIAN SELECTION PROBLEM

In order to simplify the analysis, we limit the selection problem to the case of approximating the sample median. This is slightly different than the previous problem, in that the answer must be an element of $\{X_i\}$. The elements X_i may be from an arbitrary linearly ordered set, so they may not be real numbers, or even numbers at all, and therefore it may not be possible to average them. Otherwise, Algorithm 1 would be a perfectly suitable way to approximate the median element.

As a result of this restriction, the "approximation" is really approximation of the relative rank of the desired element. We may therefore identify the elements of $\{X_i\}$ with their relative ranks, the numbers $i/(n+1)$ for $1 \le i \le n$. We seek the median element "1/2", and measure the error between the relative rank of the element produced by our algorithm and the relative rank 1/2 we desire.

The following algorithm illustrates Theorem 2:

ALGORITHM 2 (The median problem): Let $r > 1$ be an odd integer, and let $n = r^s$ for some integer $s \ge 1$. The partition size r is constant throughout, and s is the depth of recursion, or the depth of the "tree" as in Figure 1. Again, n is the size of the original set S.

```
procedure median_est(S);
begin
    set T := φ, Q;
    if |S| = 1 then return(S);
    for j := 1 until |S|/r do
        begin
            Q := next r elements of S;
            T := T U (exact median
                      element of Q)
        end;
    return(median_est(T))
end
```

The operation of the algorithm for $n = 9$, $r = 3$ is illustrated in Figure 1. This special case of the algorithm has been proposed independently by Tukey [1977], who calls the corresponding estimate the "ninther". He is more interested in the statistical, rather than the algorithmic, aspects.

Although the algorithm is presented here as a recursive procedure operating "bottom-up" to make its operation clear, it can be made space-efficient for on-line processing of S if it is implemented to operate "top-down". The time used is $O(n)$, and for the top-down implementation the space used is r locations at each level of the tree, for a total of $rs = r \log n / \log r = sn^{1/s}$. The expected relative rank of the element produced is 1/2 by symmetry considerations.

Bounding the variance of the estimate is complicated by the fact that the relative ranks of the elements produced at step (2) are not independent. However, the dependencies introduced (effectively by

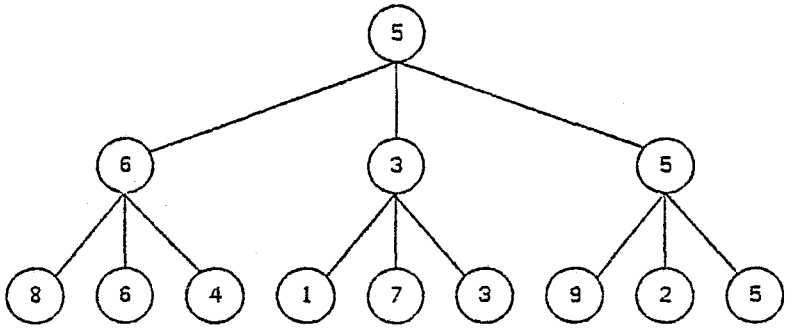


FIGURE 1 - An example of Algorithm 2 with $n = 9$ and $r = 3$. In this case, the actual median "5" is found; the algorithm always finds an element ranked 4, 5, or 6 if $n = 9$ and $r = 3$.

"sampling without replacement") tend to cluster these elements even closer to the true median than would be the case if the ranks were simply independent random variables from a uniform distribution between 0 and 1. Thus, a bound on the variance should be obtained by considering this easier case.

Carrying through this analysis gives a bound on the variance of $(\pi/2)^s/(2\pi n)$, where s is the depth of the tree. Expressing s in terms of n and r and rewriting gives the bound stated in Theorem 2, $O(n^{-1} + \log(\pi/2)/\log r)$. Simulation results tend to confirm that the actual variance is in fact bounded by this function and, even for small n and r, decreases at about this rate (see Figures 2 and 3).

As an example, assuming that the relative ranks are uniformly distributed between 0 and 1, let $r = 99$ and $n = 99^3$. In linear time, and space < 300, we can find an element whose relative rank is between .492 and .508 with probability at least 0.99 (using the Chebychev inequality). Assuming the asymptotic normal distribution for the ranks which results if they are considered to be uniformly distributed on (0,1), the estimate is between .4992 and .5008 with probability at least 0.99. Interactions among the ranks reduce this interval even further.

IV. CONCLUSIONS

Two algorithms are presented which show that space efficiency need not be poor for the order statistics problem or for the median problem. The fact that asymptotically no accuracy is lost for the former problem, and that very little is lost for the latter, indicates that sacrificing a small amount of accuracy can go a long way toward saving space. This phenomenon has already been amply demonstrated for accuracy vs. time by the many approximation algorithms which have appeared recently in the literature. By using sampling algorithms, we can also show tremendous time and space savings at only a slight cost in accuracy.

Acknowledgements - The author is indebted to Michael Shamos for first suggesting the idea of space-efficient on-line selection algorithms. Numerous discussions with Bill Eddy were also very helpful. Jay Kadane and John Lehoczky pointed out the weak conditions sufficient for the proof of Theorem 1.

V. REFERENCES

David, H.A. Order Statistics. Wiley and Sons, New York, 1970.

Knuth, D.E. The Art of Computer Programming, Vol. III: Sorting and Searching. Addison-Wesley, Reading, MA, 1973.

Sen, P.K. On the moments of the sample quantiles. Calcutta Stat. Assoc. Bulletin 9, 33 (Sept. 1959), 1-19.

Tukey, J., 1977, to appear.

Walker, A.M. A note on the asymptotic distribution of sample quantiles. J. Royal Stat. Soc. 30, 3 (1968), 570-575.

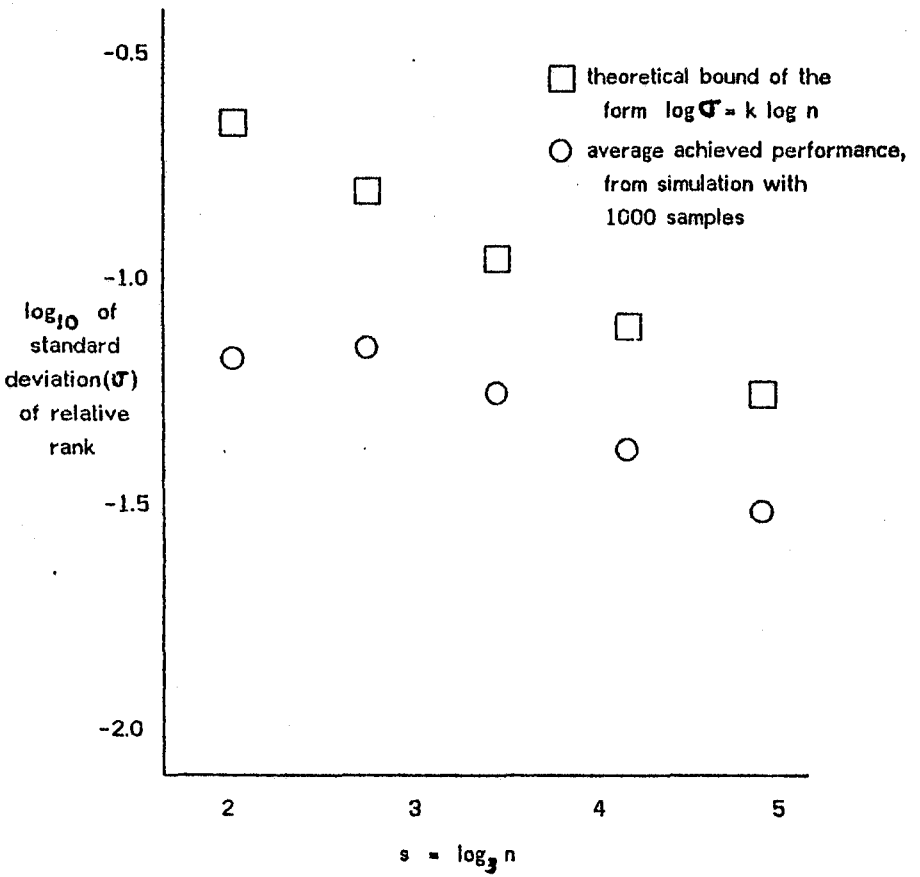


FIGURE 2 - Results for Algorithm 2 with r = 3 and various values of s

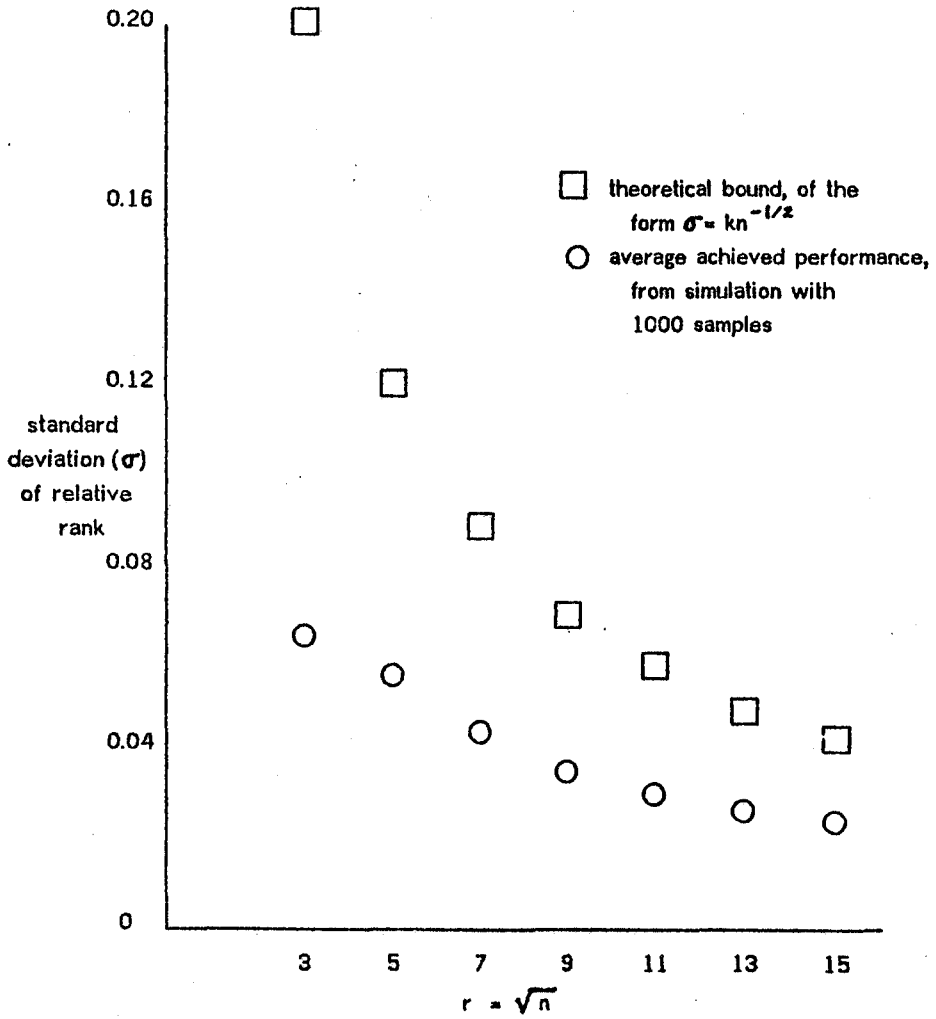


FIGURE 3 - Results for Algorithm 2 with s = 2 and various values of r.

CONTRIBUTIONS BY PAPER

Statistical Methods in Computer Performance Evaluation: A Binomial Approach to the Comparison Problem

Paul D. Amer and Sandra A. Mamrak
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

Abstract

Sophisticated improvements in both computer hardware and software have made the task of analyzing system performance increasingly more complex. Many analyses undertaken during the lifetime of a computer system require experimenters to compare alternatives: either different systems for purchase or different improvements for tuning a single system. These comparisons are performed in order to determine 'which alternative is the best'. Unfortunately, due to a general lack of performance theory, comparison experiments tend to be unstructured and wasteful. What is needed is a set of general, but precisely defined, procedures which can be employed in the task of evaluating system performance. One approach for developing these procedures is that of interfacing established statistical methodology (in particular, the theory of ranking and selection) with experimental design for computer comparisons. Two procedures have already been developed [MAM77] and another, referred to as a binomial approach, is presented here. Further efforts to interface statistical theory and computer performance evaluation will provide the theoretical resources needed by performance analysts to properly design comparison experiments.

1.0 INTRODUCTION

Sophisticated improvements in both computer hardware and software have made the task of analyzing system performance increasingly more complex. Simple comparison of execution times of various mathematical operations is no longer sufficient for supplying the bulk of information in a comparative analysis. Factors such as multiprogramming, virtual memory, intelligent terminals and master-slave computer 'teams' are just a few of the complications which have increased today's lack of understanding as to why a system behaves the way it does. In order to adjust for this increasing complexity, performance analysts often collect massive amounts of data, much of which is left unused. Data analysis is then commonly structured around the already existing data, rather than flowing smoothly from a statistically sound experimental design. Certainly this is unacceptable. What is needed is a set of general, but precisely defined, procedures which can be employed in the task of evaluating system performance.

This paper is concerned with the establishment of one of these procedures through the interface of computer performance studies with a well developed area of statistical theory. It has been divided into three major sections. The first (Section 2) is a general motivation for the application of statistical techniques to computer performance experiments. It describes the comparison theme which is central to many performance studies and how the application of statistical techniques can help reduce some of the current problems existing in the comparison environment. Section 3 is a description of statistical ranking and selection theory and its relationship to computer comparison studies. Included is an account of the initial work interfacing the two areas entitled, "Statistical Methods for Comparison of Computer Services" [MAM77]. Section 4 is an introduction and analysis of a binomial approach to the computer comparison problem. This new approach takes advantage of ranking and selection of binomial populations -- something not previously attempted. Section 4 is followed by a brief summary of the results with a motivation for continued research efforts along these lines.

2.0 COMPARISON STUDIES

Computer performance experiments often deal with the comparison of a set of alternatives. Throughout a system's lifetime, the theme of comparison is present. Unfortunately most comparison efforts lack the structure necessary for the results to be of maximum use to the experimenter. One solution to this problem is to design experiments in such a way that statistical methodology which has been previously developed can provide the needed structure.

2.1 Comparison Theme in Design-Purchase-Tuning Stages

A computer system's existence may be divided into three stages: design, purchase and tuning [GRE72]. This characterization provides a good basis for seeing how evaluation experiments, performed during different times of a system's lifetime, can be improved through interaction with statistical theory.

The design stage begins with the initial ideas of creation and lasts until the completed product is ready for marketing. Basic performance questions here include compatibility with existent machinery and superiority over competitor's equipment. It is desirable, although extremely difficult, to compare proposed ideas in order to develop the best product possible. Any company which ignores careful design performance analysis is likely to find its products the unfortunate losers when buyers perform their purchase comparison experiments later on.

The purchase stage begins with the initial salesperson/prospective-buyer interactions and lasts

until the system has been contracted and installed. Benchmark testing, cost-benefit analyses and other empirical comparisons are carried out by selection committees in the process of choosing what is generally a $10,000 to $1,000,000+ purchase. Experimentation is most important in order to determine the 'best' alternative to purchase since future performance improvements are inherently limited by the system chosen during this stage.

Once a system has been installed, in many cases a team of system analysts begin an on-going battle to improve the machine's performance. This stage, commonly referred to as a tuning period, includes testing and comparison of alternative scheduling algorithms, priority classifications and new software. Constant experimentation of this nature continues throughout a system's lifetime with the hopes of bettering its performance capabilities.

In all three stages of development, a central theme is apparent: the need for comparison. Whether the goal is to compare ideas for design, available systems for purchase or proposals for improving performance, the basic desire is to evaluate the relative merits of a set of possible choices. Given several such alternatives, an experimenter wants to know 'which one is the best?'. This question provides the foundation on which to base an experimental design. One natural approach is to 1) determine a criterion (criteria) by which the alternatives are to be judged, 2) collect measurements which represent the criterion (criteria), and 3) select that alternative which results in the optimal measured values. Unfortunately, it is not always clear how to successfully accomplish the first and last steps in such a way that the overall experiment has an associated statistical confidence statement. One potential solution to this problem is discussed in Section 3. At this point, a closer look at how interactions with established statistical theory can improve comparison studies is appropriate.

2.2 Comparison Environment

Comparison experiments are commonly laced with potential inaccuracies due to constrained budgets. In the case of a system procurement, costs which can accrue from a complete pre-purchase analysis may be as much as one-third of the actual system's cost. It is not difficult to understand, therefore, why only partial analyses are performed with less than optimal selections resulting. Part of the reason for such high costs in comparison efforts is the lack of available performance theory on which analysts can design their experiments. Every time a need for comparison arises, whether it be in private business, academic communities or government installations, personnel time and effort must be spent devising an experiment which is appropriate for the particular comparison problem at hand. Having little performance theory to use as a guide, the analysts often produce loosely structured experiments based on what a few people judge is 'reasonable'. Also, since most procurement data is proprietary, there is little published literature describing past comparison efforts. Given an identical comparison problem and criteria for selection, seldom will two performance groups initiate identical experiments or build upon formerly used techniques.

As a solution to this lack of consistency, it is proposed that the development of a package of general and reliable experimental procedures be undertaken. Each one will provide three parts: 1) a description of the different computer environments for which the procedure is applicable, 2) an experimental design indicating the necessary data collection and statistical computation steps and 3) an explanation of the significance of the results. From this package of procedures, performance experts could select the procedure most appropriate for their particular comparison problem. The results of establishing such a package would be improved, more structured comparison efforts, more confidence in experimental results and decreased experimental costs.

The development of a set of designs with such favorable characteristics is feasible through efforts to interface statistical theory with the area of computer comparison. An initial effort to begin such an interface is described in [MAM77]. In that work (discussed in Section 3), statistical theory of ranking and selection was successfully applied to the comparison problem. A second effort is the binomial approach presented in Section 4. Continuing efforts in this area, aimed at meeting the goal of a package of widely applicable experimental comparison procedures, will remove many of the obstacles that exist in the current comparison environment due to lack of available theory.

A detailed description of the positive benefits available from the application of statistical theory to the performance evaluation field was published several years ago by Grenander and Tsao [GRE72]. In their state of the art appraisal they declare, 'There is no doubt that the task of evaluating the performance of computer system is of utmost importance, both to computer manufacturers and to users of computers. It deserves a more systematic study than it has received until now and we believe that modern statistical methodology offers powerful tools that have not yet been exploited to their full capacity for this purpose.' The establishment of a set of computer comparison procedures derived from powerful statistical tools, one of which is ranking and selection theory, will provide support for their prediction.

3.0 RANKING AND SELECTION

3.1 Relation to Computer Comparison Problem[†]

When judging different systems, it is desirable to rank the alternatives in some numerical fashion and assert the system with the superior ranking (largest or smallest, depending on the criterion being evaluated) as the best one. An obvious technique for deriving a numerical ordering based on a single criterion is to perform an empirical analysis which results in a single value for each system under consideration. For maximum credibility, it is desirable to design the experiment in a manner which provides a confidence statement concerning the probability that the best observed value truly comes from the best alternative. Since experimental results are variable, there always exists the possibility that an inferior system has a 'good day' while the best system performs below normal. The goal of a good experimental design should be to know precisely what the statistical chances are of such a nonrepresentative sampling. The theory of ranking and selection offers the computer performance analyst an excellent foundation for satisfying this goal.

In general, ranking and selection considers a set of populations, each with a probability distribution. Sample observations are collected from each population and a single statistical estimate such as the mean or α-quantile is computed. These estimates are ranked in numerical order and the population associated with the superior one is labeled as being the best. A variety of assumptions concerning the populations and the data collection process can be made. These include assumptions regarding the underlying form of the distribution (e.g., normal, exponential, etc.) and the level of dependence between obtained observations. An excellent summary of the numerous ranking and selection techniques which are available can be found in [GIB77].

3.2 Mamrak and DeRuyter Study

The initial effort in applying ranking and selection theory to computer comparison studies was performed by Mamrak and DeRuyter [MAM77]. In that study, the authors present two computer comparison methodologies related to ranking and selection techniques. One is for ranking populations by sample means

[†]For purposes of clarity, the discussion of comparison efforts will be in terms of rating alternative computer systems (purchase stage). However, the methodologies are equally applicable to the comparison of performance improvements on a single system (tuning stage). In such a situation, each improvement on the single system is analogous to a completely separate system.

assuming that the distributions are normal and the observations collected are independent.† It is not often the case, however, that a normality assumption is valid for computer performance data. The other methodology is for ranking according to sample α-quantiles ($0<\alpha<1$) where no assumption is made about the underlying form of the distributions and again the observations are independent.

In both cases, the experiment is designed so that the probability of correctly selecting the superior population is predetermined by the experimenter to be any value between 1/k and 1, where k is the number of populations being compared (1/k is the lower bound since random selection guarantees at least that value). A case study was performed in which the populations compared were response time distributions of two interactive services, thus verifying the practicality of the two techniques. By interfacing ranking and selection theory with the field of computer performance evaluation, the work offers an experimenter two new procedures for performing computer comparisons. More important, the comparison procedures are well-defined for easy implementation and have an associated confidence statement regarding their precision.

There still remain many situations, however, for which the procedures are not appropriate. The context in which ranking and selection techniques were discussed in the Mamrak and DeRuyter study was that of comparison of computer services available by remote terminal access. (For purposes of clarity, the binomial approach which will be introduced and discussed later in the paper will be presented in the same context. Its use, however, is extendible to other comparison environments.) In their examination of computer services, the authors note that 'the general nature of computer performance data (a large number of relatively small values and small number of large values) has led many analysts to express their expectations in terms of α-quantiles'. Based on their methodology for ranking according to α-quantiles, results are produced of the form:

> If computer service A has 90% of its response times less than 3 seconds,
> and computer service B has 90% of its response times less than 3.5 seconds,
> then rank A as being better than B.

In this case the criteria for establishing the superior system is response time. Other criteria, herein denoted Φ, which may be appropriate under different circumstances, are listed in Table 1. Stated more precisely, the procedure involves estimating the distribution functions of k populations at a predetermined α-quantile. Let $F_k \preceq F_{k-1} \preceq \cdots \preceq F_1$ denote the distribution functions ranked according to their α-quantile values, $x_\alpha(F_i)$,

$$\text{i.e., } x_\alpha(F_k) \leq x_\alpha(F_{k-1}) \leq \cdots \leq x_\alpha(F_1)$$

The procedure examines the difference between the distribution with the smallest α-quantile value, F_k, and all of the other distributions over a small interval around $x_\alpha(F_k)$ denoted $I = [x_{\alpha-\varepsilon}(F_k), x_{\alpha+\varepsilon}(F_k)]$. (See Figure 1). As long as a minimum distance, $d^*>0$, exists between F_k and F_{k-1} throughout this interval, the number of observations, n, required to statistically guarantee a given probability of correctly selecting the best population on the basis of empirical results can be precisely calculated. The distance value d^* may be determined by considering that any loss incurred by an incorrect selection of an alternative whose distribution lies closer than d^* to F_k within the noted interval is insignificant. For practical purposes all such alternatives are considered equivalent to the best one. Tables of n for a few particular combinations of k, P^*, d^*, ε and α were computed in the original ranking and selection research

†A population is the same as the distribution of a particular variable (criterion) on a system, e.g., distribution of response times on an interactive system.

TABLE 1. Measures Useful for the Comparison of Computer Services [ABR77]

NAME	DESCRIPTION
1. System delay	Elapsed time from end of user input to beginning of system output
2. System transit	Printout or output display time
3. Acknowledgement delay	Time from input carriage return to first system reaction
4. User transmit	Time for user typing and transmission
5. Interprocess transfer	Time to move a process between different software or hardware processors
6. Total session	Time for complete interactive session
7. Character arrival rate	Number of characters per unit time, averaged over the entire session
8. Task completion	Individual user task throughput rate
9. Total error	Number of errors per unit time; a reliability measure
10. User idleness	Fraction of time the user is waiting
11. CPU time	Total job processing time
12. Throughput	Number of completed jobs per unit time

[SOB67]. Very extensive tables are currently being generated by Dudewicz and Dhariyal [DUD78] at The Ohio State University. Some were given in [MAM77].

3.3 Binomial Approach

In practice, it is uncommon that an experimenter will know which α-quantile for a particular Φ is appropriate in a computer comparison. Such knowledge is important, since as indicated in Figure 1, comparing distributions at different α-quantiles may result in different orderings. It is more likely the case that an experimenter has a criterion and particular threshold value in mind, say Φ_{THLD}, and wishes to determine the system which has the largest proportion of Φ observed values less than (or greater than) Φ_{THLD}. This is often true of response time at most user terminals, as illustrated in Example 1.

Example 1

> Psychological studies have indicated that excessive and unpredictable delays on a time sharing system result in a degradation of user performance [CAR68]. In particular, it is claimed that response delays of 15 seconds or longer rule out conversational interaction between human and information systems [MIL68]. Hence a purchaser of an interactive system would find it most profitable to buy a system which provides the largest proportion of response times less than 15 seconds and is still within the limitations of other constraints such as cost and available services.

This particular view of the comparison problem can be expressed in such a manner that a segment of ranking and selection theory thus far unused in computer comparison experiments becomes applicable. The general goal of the 'binomial approach' is to determine:

(*) Which system has the largest proportion (or percentage) of its response time (Φ) values less than a threshold value Φ_{THLD}.

Using the above goal, results analogous to those which Mamrak and DeRuyter derived would be of the form:

> If computer service A has 80% of its response times less than Φ_{THLD} and computer service B has 87% of its response times less than Φ_{THLD}, then rank B as being better than A.

It is important to note the difference in the two approaches. In the Mamrak and DeRuyter procedure, the α-quantile ($0<\alpha<1$) is predetermined and the best system is the one with the optimal x_α value for some criterion Φ. In the binomial approach, a threshold value, Φ_{THLD}, is chosen and the best system is the one with the largest proportion, $p (0<p<1)$, of Φ values less than Φ_{THLD}. In a sense, α is analogous to p and x_α is analogous to Φ_{THLD}. A comparison of Figures 1 and 2, along with the explanation in Section 4, will help to clarify this distinction further.

4.0 BINOMIAL APPROACH ANALYSIS

4.1 Relation to Computer Comparison Problem

By treating the comparison question as in (*), work performed on comparing probabilities of success of several binomial distributions [SOB57] takes on major importance in performance studies. A good analogy for understanding the significance of the binomial distribution research is found in an example of comparative life testing of light bulbs.

Example 2

> Suppose a company representative has a set of alternative brands of bulbs from which to choose. Since the size of the purchase is considerable, the buyer wishes to first test the bulbs in an attempt to determine which brand is the best. It is decided that the best brand will be the one with the largest proportion of bulbs which operate at least 1000 hours. In other words, the company wants those bulbs which have the highest probability of maintaining 1000 hours or more of service.

In the above example the criterion, Φ, is life expectancy and Φ_{THLD} is 1000 hours. Any bulb which lasts longer than 1000 hours is considered a success unit. Φ_{THLD} is chosen according to non-statistical considerations and is held constant throughout the experiment. Each brand of bulbs has an associated life expectancy probability distribution, f_i, and cumulative distribution, F_i. (See Figures 2a,b.) In this case the goal of any experiment is to determine the brand for which $F_i(\Phi_{THLD})$ is a minimum. For the hypothesized distributions in Figure 2b, it can be seen that 60% and 70% of Brands 2 and 3, respectively, go bad by 1000 hours whereas only 50% of Brand 1 do likewise. Hence Brand 1 is the best since half of its bulbs last longer than the 1000 hour threshold value, more than either of the other 2 brands.

The problem with the above analysis is that the probability and cumulative distributions of the different alternative brands are not known a priori and therefore must be estimated by some form of empirical study. Specifically it is desired to estimate the cumulative distributions at the single value Φ_{THLD}, i.e., $F_i(\Phi_{THLD})$, for $i = 1, \ldots, k$ where k is the number of brands being compared. One way to go about this is to test n bulbs selected at random from each of the k brands and let X_i be the number of bulbs collected from the i^{th} brand which lasts less than 1000 hours. Then X_i has a binomial distribution with parameters n and p_i where p_i is the probability a bulb from brand i will not last longer than 1000 hours. Note that p_i is identical to $F_i(\Phi_{THLD})$ and the value $\bar{X}_i = X_i/n$ is an unbiased estimator for that particular population parameter.

An analogous situation in computer studies was given in Example 1 in Section 3. In that example, the analyst is interested in determining which interactive system has the largest proportion of response times less than 15 seconds. By collecting n random response times from each system and letting X_i be the number of responses less than 15 seconds, then X_i is binomial (n,p_i) with $\bar{X}_i = X_i/n$ as an estimator for p_i. As a result of considering each of the collected observations as being in only one of two states, < 15 secs. or $\geq$ 15 secs., information regarding exact measurements is discarded. However, the experiment is designed so that existent ranking and selection theory becomes applicable.

One problem is determining how many response time observations need to be collected in order that the experimental estimates of p_i (and likewise $F_i(\Phi_{THLD})$) are accurate enough to satisfy an overall probability confidence statement. This is where Sobel's work on ranking binomial populations according to probability of success becomes important.

4.2 Selection of the Best Binomial Population [SOB57]

In the following discussion there are $k \geq 2$ computer systems from which it is desired to select the best one according to a single criterion, Φ. (Recall that these results are equally applicable to $k \geq 2$ alternatives on a single system.) One must collect a sufficient amount of data in order to statistically guarantee that the probability of making a Correct Selection (CS) is at least a specified value P*, $(\frac{1}{k} < P^* < 1)$, i.e., $P(CS) \geq P^*$. Observations from the same system have a common probability of being less than a threshold value, Φ_{THLD}. Let $p_{[1]} \leq p_{[2]} \leq \cdots \leq p_{[k]}$ denote the ordered p_i values where the assumption is made that the experimenter has no a priori knowledge about the correspondence between the ordered $p_{[i]}$ and the k identifiable populations (systems) which are themselves denoted π_i, $i = 1, 2, \ldots, k$. The goal (*) is then reduced to:

(**) Select π_i which has associated $p_{[k]}$ where $P(CS) \geq P^*$.

The exact mathematical solution to this problem employs what is known as an 'indifference zone' approach. The indifference zone is a minimum distance, $d^* > 0$, between the best probability, $p_{[k]}$, and the remaining others, which the experimenter feels it is significant to be able to detect. That is, it is felt that any loss incurred by incorrectly selecting a population whose p_i value is less than $p_{[k]}$, but not more than d^* less, is insignificant. This is similar, but not exactly the same as the meaning of d^* described in Section 3. As d^* decreases, more observations are required in order to increase the accuracy with which the p_i values are estimated. A smaller d^* implies that the experimenter wishes to be able to detect a finer difference among the populations being compared. The complete experimental goal is now:

(***) Select π_i which has associated $p_{[k]}$ where $P(CS) \geq P^*$ whenever $p_{[k]} - p_{[k-1]} \geq d^*$.

The derivation of the mathematical solution and extensive tables and graphs of required n for certain values of k, d^* and P^* can be found in [SOB57]. A few selected tables have been reproduced in Appendix A.

Example 3

> Suppose there are four priority schemes for processing the transactions which are transmitted from various user terminals. A transaction can be considered to be any user request for processing, usually initiated upon striking the return key of the keyboard. The priority schemes are to be tested in an attempt to determine the 'best' one for permanent installation. The deciding factor to be used in judging the best one is: 'which system processes the largest proportion of individual user transactions in less than 15 seconds'. This criterion might be appropriate in an environment where the major concern is avoiding extremely long response times at terminals, such as in a bank or reservation system (airline, hotel, etc.). It is desired that the experiment produce at least 90% confidence of correctly selecting the best priority scheme whenever the difference between the largest and next largest proportion is at least .10.

Figure 1. Illustration of the Distance Measure for α-Quantiles (MAM77)

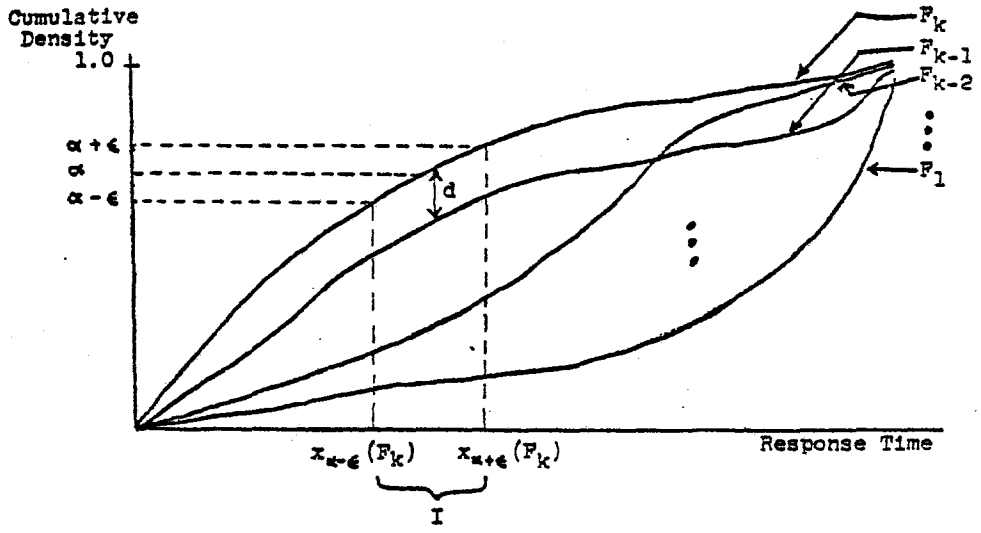


Figure 2a. Bulb Life Expectency Probability Distribution.

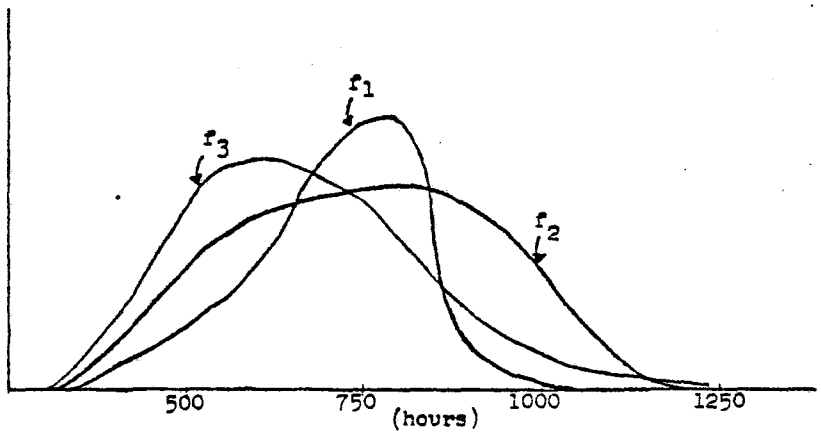
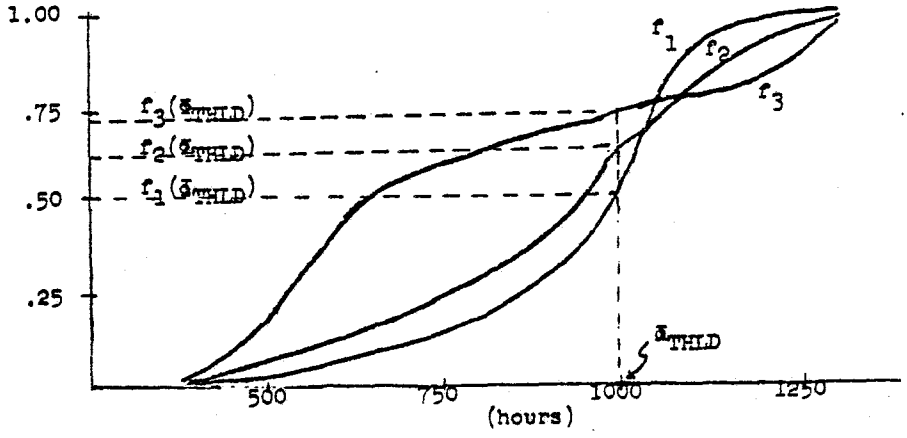


Figure 2b. Bulb Life Expectency Cumulative Distribution.



That is, as long as the best priority scheme processes ten percent more 15-second-or-less transactions than any of the others, the results of the empirical study are statistically guaranteed to determine which scheme is the best at least 90% of the time. That is given $k = 4$, $d^* = .10$, $P^* = .90$, $P(CS) \geq .90$ whenever $p_{[4]} - p_{[3]} \geq .10$.

Simple table look-up (See Table III in Appendix A) confirms that response times from 150 random user transactions must be collected from the system under each priority scheme in order to satisfy the stated requirements.

4.3 Experimental Design

Step 1. Collect n independent Φ observations from each of the k alternatives in question, where n is determined by table look up based on k and the P* and d* values supplied by the analyst.

Step 2. Let X_i = number of observations from alternative i which are $\leq \Phi_{THLD}$ (also supplied by the analyst).

Step 3. Compute $\bar{X}_i = X_i/n$, the estimates of the p_i.

Step 4. Select the alternative which produced the largest $\bar{X}_i$. Note that since n is identical for all systems, selection of the alternative which produced the largest X_i has the same results. In the case of ties for the largest $\bar{X}_i$ (which is not completely unlikely due to the common sample size), the probability statement will be satisfied by making a random selection among the alternatives whose $\bar{X}_i$ values were tied.

Comment: One should note that following the steps outlined above guarantees P* as a minimum probability value for a correct selection. This is true since n is derived assuming the worst possible combination of p_i values. This combination is known as the least favorable configuration. (For a discussion of l.f.c., see [GIB77] - Section 1.3 or [KLE75] - Section V.C.2). In reality it is seldom that such a configuration actually exists, but since the true arrangement is unknown, an exact numerical solution must be pessimistic in nature. Hence, although the experimenter selects a minimum P* value, the actual experiment is likely to have an even larger probability of correct selection.

5.0 CONCLUSION

The computer system life cycle is divided into three stages - design, purchase and tuning. From a performance evaluation viewpoint, the theme of comparison is present in all three stages. Current comparison efforts in the purchase and tuning stages appear to be unstructured and at times inaccurate due to practical cost constraints. It is proposed that further advances in interfacing the area of statistical theory with the experimental designs of comparison effects will improve these efforts. Establishment of a collective set of statistically sound procedures, from which performance analysts can choose to suit their particular problem, appears to be one potential approach to such an interface. Mamrak and DeRuyter offered two comparison procedures which employed ranking and selection theory. A binomial approach procedure is presented here as another general experimental design to answer a different class of comparison problems. As with the Mamrak and DeRuyter procedures, its presentation is in the context of comparison of computer services, yet it is also applicable in any performance environment in which selection of the alternative with the largest proportion of criterion values less than a certain threshold value is desired.

6.0 BIBLIOGRAPHY

[ABR77] Abrams, M. D. and S. Treu. "A Methodology for Interactive Computer Service Measurement," Communications of the ACM, Vol. 20, No. 12, December 1977, pp. 936-944.

[CAR68] Carbonell, J. R., J. Elkind and R. Nickerson. "On the Psychological Importance of Time in a Time Sharing System," Human Factors, Vol. 10, April 1968, pp. 135-142.

[DUD78] Dudewicz, E. J. and I. D. Dariyal. "Sobel's Nonparametric Selection Procedures for Quantiles: Extension and Tables," Technical Report, Department of Statistics, The Ohio State University, Columbus, Ohio, to appear.

[GIB77] Gibbons, J. D., I. Olkin and M. Sobel. Selecting and Ordering Populations: A New Statistical Methodology. New York: John Wiley and Sons, 1977.

[GRE72] Grenander, V. and R. F. Tsao. "Quantitative Methods for Evaluating Computer System Performance: A Review and Proposals," Statistical Computer Performance Evaluation, edited by W. Freiberger. New York: Academic Press, 1972, pp. 3-24.

[KLE75] Kleijnen, J.P.C. Statistical Techniques in Simulation, Part II. New York: Marcel Dekker, Inc., 1975.

[MAM77] Mamrak, S. A. and P. A. DeRuyter. "Statistical Methods for Comparison of Computer Services," Computer, Vol. 10, No. 11, November 1977, pp. 32-39.

[MIL68] Miller, R. B. "Response Time in Man-Computer Conversational Transactions," AFIPS Conference Proceedings, Vol. 33, Part 1, 1968 FJCC, pp. 267-277.

[SOB57] Sobel, M. and M. J. Huyett. "Selecting the Best One of Several Binomial Populations," Bell System Technical Journal, Vol. 36, March 1957, pp. 537-576.

[SOB67] Sobel, M. "Nonparametric Procedures for Selecting the t Populations With the Largest α-Quantiles," Annals of Mathematical Statistics, Vol. 38, 1967, pp. 1804-1816.

APPENDIX A: Tables for Selecting the Best of k Binomial Populations [SOB57]

TABLE I — NUMBER OF UNITS REQUIRED PER PROCESS TO GUARANTEE A PROBABILITY OF P^* OF SELECTING THE BEST OF k BINOMIAL PROCESSES WHEN THE TRUE DIFFERENCE $p_{(1)} - p_{(2)}$ IS AT LEAST d^*. ($k = 2$)

The three values in each group are: (1) Normal approximation, (2) Straight line approximation, and (3) Smallest integer required.

d^*	P^*							
	0.50	0.60	0.75	0.90	0.95	0.96	0.95	0.99
0.05	0	12.51	90.77	141.30	211.29	327.66	539.77	1079.70
	0	12.84	90.99	141.66	214.53	328.48	541.12	1082.41
	0	14	92	142	215	329	541	1082
0.10	0	3.13	22.52	35.06	53.17	81.30	133.93	267.90
	0	3.21	22.75−	35.41	53.71	82.12	135.28	270.60
	0	4	23	36	54	83	135	270
0.15	0	1.39	9.88	15.39	23.33	35.68	58.78	117.57
	0	1.43	10.11	15.74	23.87	36.50−	60.12	120.27
	0	2	11	16	24	37	60	120
0.20	0	0.77	5.16	8.50−	12.89	19.71	32.47	64.94
	0	0.80	5.60	8.88+	13.43	20.53	33.82	67.65+
	0	1	6	9	14	21	34	67
0.25	0	0.48	3.41	5.31	8.06	12.32	20.23	40.39
	0	0.51	3.64	5.67	8.59	13.14	21.64	43.20
	0	1	4	6	9	14	22	42
0.30	0	0.32	2.30	3.53	5.43	8.30	13.68	27.36
	0	0.36	2.53	3.93	5.97	9.12	15.03	30.07
	0	1	3	4	6	9	15	29
0.35	0	0.23	1.63	2.54	3.85−	5.88	9.69	19.33
	0	0.26	1.86	2.89	4.38	6.70	11.04	22.09
	0	1	2	3	5	7	11	21
0.40	0	0.17	1.19	1.86	2.82	4.31	7.10	14.21
	0	0.20	1.42	2.21	3.36	5.13	8.46	16.91
	0	1	2	3	4	5	9	16
0.45	0	0.13	0.90	1.39	2.11	3.23	5.33	10.65+
	0	0.16	1.12	1.73−	2.65+	4.06	6.68	13.36
	0	1	2	2	3	4	7	13
0.50	0	0.10	0.68	1.06	1.61	2.46	4.06	8.12
	0	0.13	0.91	1.42	2.15−	3.28	5.41	10.82
	0	1	1	2	3	4	5	10

TABLE II — NUMBER OF UNITS REQUIRED PER PROCESS TO GUARANTEE A PROBABILITY OF P^* OF SELECTING THE BEST OF k BINOMIAL PROCESSES WHEN THE TRUE DIFFERENCE $p_{(1)} - p_{(2)}$ IS AT LEAST d^*. ($k = 3$)

The three values in each group are: (1) Normal approximation, (2) Straight line approximation, and (3) Smallest integer required.

d^*	P^*							
	0.50	0.60	0.75	0.90	0.95	0.96	0.95	0.99
0.05	30.89	78.16	205.06	272.38	363.06	496.14	732.63	1305.21
	30.97	78.36	205.58	273.04	363.97	497.38	734.16	1308.40
	31	79	206	273	364	498	735	1308
0.10	7.66	19.39	50.88	67.38	90.08	123.10	181.78	323.55+
	7.74	19.59	51.39	68.26	90.99	124.34	181.62	327.12
	8	20	52	69	91	125	184	327
0.15	3.38	8.511	22.37	29.66	39.53	54.02	79.77	142.12
	3.44	8.71	22.84	30.34	40.44	55.26	81.61	145.39
	4	9	23	31	41	55	82	145
0.20	1.86	4.70	12.33	16.38	21.84	29.84	44.07	78.51
	1.94	4.90	12.85−	17.07	22.75−	31.09	45.90	81.78
	3	5	13	17	23	31	46	81
0.25	1.16	2.94	7.71	10.21	13.65−	18.65+	27.54	49.07
	1.24	3.13	8.22	10.92	14.56	19.90	29.38	52.34
	2	4	9	11	15	20	29	52
0.30	0.78	1.98	5.20	6.90	9.20	12.57	18.57	33.08
	0.86	2.18	5.71	7.58	10.11	13.82	20.40	36.35−
	2	3	6	8	10	14	20	35
0.35	0.53+	1.40	3.68	4.89	6.52	8.91	13.15+	23.43
	0.63	1.60	4.20	5.57	7.43	10.15	14.99	26.70
	2	2	5	6	8	10	15	26
0.40	0.41	1.03	2.70	3.58	4.78	6.53	9.64	17.17
	0.48	1.22	3.21	4.27	5.69	7.77	11.48	20.45−
	1	2	4	5	6	8	11	20
0.45	0.30	0.77	2.02	2.69	3.58	4.90	7.23	12.88
	0.38	0.97	2.54	3.37	4.49	6.14	9.07	16.15+
	1	2	3	4	5	6	9	15
0.50	0.23	0.59	1.54	2.05−	2.73	3.73	5.51	9.81
	0.31	0.78	2.06	2.73	3.64	4.97	7.34	13.08
	1	2	3	3	4	5	7	12

APPENDIX A (continued)

TABLE III — NUMBER OF UNITS REQUIRED PER PROCESS TO GUARANTEE A PROBABILITY OF P^* OF SELECTING THE BEST OF k BINOMIAL PROCESSES WHEN THE TRUE DIFFERENCE $p_{(1)} - p_{(2)}$ IS AT LEAST d^*. ($k = 4$)

The three values in each group are: (1) Normal approximation, (2) Straight line approximation, and (3) Smallest integer required.

d^*	P^*							
	0.50	0.60	0.75	0.90	0.95	0.96	0.95	0.99
0.05	69.85−	132.65+	252.27	357.32	496.82	699.53	818.30	1438.12
	70.02	132.99	252.98	358.42	497.96	701.03	820.42	1441.72
	71	134	253	359	498	601	850	1442
0.10	17.33	32.91	70.04	88.71	113.35−	148.76	210.48	356.83
	17.51	33.25−	70.74	89.61	114.49	150.26	212.61	360.43
	18	34	71	90	115	150	212	360
0.15	7.61	14.44	30.74	38.93	49.74	65.29	92.37	156.61
	7.78	14.73	31.44	39.82	50.88	66.73	94.49	160.19
	8	15	32	40	51	67	94	160
0.20	4.20	7.98	16.98	21.51	27.48	36.06	51.03	86.50+
	4.33	8.31	17.69	22.40	28.62	37.56	53.15+	90.12
	5	9	18	23	29	38	53	89
0.25	2.63	4.99	10.61	13.44	17.17	22.54	31.89	54.06
	2.80	5.32	11.32	14.34	18.32	24.04	34.02	57.67
	3	6	12	14	18	24	34	57
0.30	1.77	3.36	7.15+	9.06	11.58	15.19	21.50−	36.44
	1.95−	3.69	7.86	9.96	12.72	16.70	23.62	40.05−
	3	4	8	10	13	17	23	39
0.35	1.25+	2.38	5.07	6.42	8.20	10.76	15.23	25.82
	1.43	2.71	5.77	7.31	9.35−	12.27	17.36	29.42
	2	3	6	7	9	12	17	28
0.40	0.92	1.75−	3.71	4.70	6.01	7.89	11.16	18.92
	1.09	2.08	4.42	5.60	7.16	9.39	13.29	22.53
	2	3	5	6	7	9	13	21
0.45	0.69	1.31	2.79	3.53	4.51	5.92	8.37	14.19
	0.86	1.64	3.49	4.42	5.65+	7.42	10.51	17.80
	2	2	4	5	6	7	10	17
0.50	0.53	1.00	2.12	2.69	3.43	4.51	6.38	10.81
	0.70	1.33	2.83	3.58	4.55	6.01	8.50+	14.42
	2	2	3	4	5	6	8	13

TABLE IV — NUMBER OF UNITS REQUIRED PER PROCESS TO GUARANTEE A PROBABILITY OF P^* OF SELECTING THE BEST OF k BINOMIAL PROCESSES WHEN THE TRUE DIFFERENCE $p_{(1)} - p_{(2)}$ IS AT LEAST d^*. ($k = 10$)

The three values in each group are: (1) Normal approximation, (2) Straight line approximation, and (3) Smallest integer required.

d^*	P^*							
	0.50	0.60	0.75	0.90	0.95	0.96	0.95	0.99
0.05	216.06	312.31	511.15+	604.04	722.50−	887.54	1165.49	1795.01
	217.50+	313.29	512.43	605.53+	724.31	889.77	1168.41	1802.51
	218	314	513	606	725	890	1169	1803
0.10	53.53	77.54	126.53	149.87	179.27	220.22	289.15	446.12
	54.33	78.32	128.11	151.39	181.08	222.44	292.10	450.63
	55	79	128	151	181	222	291	449
0.15	23.62	34.03	55.66	65.77	78.67	96.64	126.90	195.77
	24.17	34.81	56.94	67.25	80.43	98.86	129.82	200.28
	25	35	57	67	80	98	129	198
0.20	13.03+	18.80	30.75−	36.33	43.46	53.39	70.10	108.15
	13.59	19.58	32.03	37.85−	45.27	55.61	73.03	112.66
	14	20	32	38	45	55	72	111
0.25	8.16	11.75−	19.22	22.71	27.16	33.37	43.82	67.59
	8.70	12.53	20.50−	24.22	28.97	35.59	46.74	72.10
	9	13	20	24	29	35	46	70
0.30	5.50−	7.92	12.93+	15.31	18.31	22.49	29.53	45.56
	6.04	8.70	14.23	16.82	20.12	24.72	32.46	50.07
	7	9	14	17	20	24	32	48
0.35	3.99	5.61	9.15	10.84	12.97	15.93	20.92	32.28
	4.44	6.39	10.46	12.36	14.78	18.16	23.85−	36.79
	5	7	11	13	15	18	23	35
0.40	2.85+	4.11	6.73	7.95−	9.51	11.68	15.34	23.66
	3.40	4.90	8.01	9.46	11.32	13.90	18.26	28.16
	4	5	8	10	11	13	17	26
0.45	2.14	3.08	5.05−	5.96	7.13	8.76	11.50+	17.75−
	2.69	3.87	6.33	7.48	8.94	10.98	14.42	22.25+
	3	4	6	8	9	11	14	20
0.50	1.63	2.35−	3.84	4.54	5.43	6.67	8.76	13.52
	2.18	3.13	5.12	6.06	7.24	8.90	11.68	18.03
	3	4	5	6	7	9	11	16

Gauss-Jordan vs. Cholesky

Kenneth N. Berk

Illinois State University

ABSTRACT

A simulation study was performed to compare Gauss-Jordan elimination, Gauss-Jordan elimination with choice of pivots, and the Choleski algorithm.

Two types of matrices were used. Type I has eigenvalues 2, 1, ..., 1, δ; Type II has eigenvalues 2, 2r, $2r^2$, ..., δ; δ takes 6 values: 10^{-4}, $10^{-4.5}$, ..., $10^{-6.5}$. Given a diagonal matrix D with the specified eigenvalues, the matrices to be inverted were obtained as PDP' where P is random orthogonal. Matrices of order 5, 10, 20 were used, so there were 6 x 3 = 18 matrices of each type.

There were several surprising results. Type II matrices were inverted with 1 to 1.5 more decimals, on the average, but the standard deviation was also ten times as large. For both types, Choleski is best by a fraction of a decimal, but pivoting is helpful only for Type II matrices. The number of decimals was predicted with less accuracy by the matrix condition than by the trace of the inverse correlation matrix.

April, 1977

Gauss-Jordan (Sweep) vs. Choleski

Kenneth Berk, Illinois State University

1. Introduction

Although more accurate methods are available using Householder or modified Gram-Schmidt orthogonalization [6], most statistical packages solve the least squares problem by applying Gauss-Jordan eliminations (sweeps) [2] to the normal equations (perhaps scaled to correlation form). The Gauss-Jordan procedure is very convenient in terms of adding and deleting predictor variables, and it is easily programmed.

For accuracy on a general matrix, the Gauss-Jordan procedure requires pivoting, choice of the row and column for elimination. However, based on the work of Wilkinson[7, p. 305] it is generally accepted that this is unnecessary for positive definite matrices. On the other hand, Wilkinson's work [7, p. 305] also suggests that the Choleski (square root) procedure is more accurate for positive definite matrices.

To compare the inversion accuracy of the methods, a simulation study was done to compare Gauss-Jordan elimination, Gauss-Jordan elimination with choice of pivots (at each stage, the pivot is chosen to maximize the tolerance, the ratio of a diagonal element to the value originally in that position). and the Choleski algorithm.

Two types of matrices were used. Type I has eigenvalues 2, 1, ..., 1, δ; Type II has eigenvalues 2, 2r, $2r^2$, ..., δ; here δ takes on six values, 10^{-4}, $10^{-4.5}$, ..., $10^{-6.5}$. Forming a diagonal matrix D with the specified eigenvalues, the matrices to be inverted were obtained as PDP' where P is random orthogonal. Matrices of order 5, 10, and 20 were used, so there were 6 × 3 = 18 matrices of each type.

To obtain a random orthogonal matrix of order n the Gram-Schmidt process was applied to an nxn array of normal random numbers. The normal

deviates were generated by the Marsaglia modified polar procedure [5], using uniform random numbers from the Lewis-Payne generator [4]. All computations were done on an IBM 370-145. The inversions were done in single precision and compared with a double precision inverse A used as a standard. Digits of accuracy of a computed inverse B were computed as

$$d = \log_{10}\left(\left(\sum_{i,j}(A_{ij} - B_{ij})^2 / \sum_{i,j} A_{ij}^2\right)^{1/2}\right).$$

2. Results

The results are summarized in Figure 1 and Table 3.

There are some surprising results. In particular, the results for Type I and Type II matrices are very different in terms of accuracy and in terms of variability.

Type II matrices were inverted with 1 to 1½ more digits, on the average, for the same δ. The overall means were 2.3 for Type II, 1.1 for Type I.

For Type I matrices, Cholesky was the clear winner although the margin depends on the order N of the matrix. Based on the model $d_{ij} = \mu + m_i + \delta_j + \varepsilon_{ij}$. where d_{ij} is the number of digits accuracy, m_i is effect from method i (i = 1, 2, 3), δ_j represents the effect from the jth value of δ(j = 1, ···, 6), and the ε_{ij} are assumed independent normal, mean 0, variance σ^2, 95% confidence intervals and estimates of σ are as follows:

Table 1 Results for Type I matrices

N	$\mu + m_1$	$m_2 - m_1$	$m_3 - \left(\frac{m_1 + m_2}{2}\right)$	10 d of f
5	1.37±.071	-.035±.100	.204±.087	.078
10	1.03±.021	-.056±.030	.128±.026	.023
20	.75±.034	-.048±.048	.058±.041	.037

Here m_1, m_2, and m_3 are the effects for Gauss-Jordan, Gauss-Jordan-pivoting, and Cholesky, respectively. The differences favoring Cholesky were expected but the adverse effect of pivoting is a big surprise. It was assumed that, if anything, pivoting would be helpful.

Originally, N was included in the linear model, but interactions of the other factors with N forced separate models for each N. Note that the Cholesky-Gauss-Jordan difference is smaller for large N. Note, too, the decrease in digits of accuracy with increasing N.

For the Type II matrices, it was also necessary to fit separate models for each N, not because of interactions, but because of variance heterogeneity. Using the same model as described above for Type I matrices, we get the following 95% confidence intervals and estimates of σ:

Table 2 Type II matrices

N	$\mu + m_1$	$m_2 - m_1$	$m_3 - \left(\frac{m_1 + m_2}{2}\right)$	10 d of f
5	2.45±.70	.40±.90	-.41±.86	.77
10	2.21±.23	-.05±.32	.023±.28	.25
20	2.23±.16	-.015±.22	-.003±.19	.17

To achieve sufficient resolving power to detect differences, the Type II results with N = 5 were replicated 40 times. Based on a model that includes effects of method, δ, interaction of mathod and δ, and replication, 95% confidence intervals and the estimated standard deviation are as follows:

N	$\mu + m_1$	$m_2 - m_1$	$m_3 - \left(\frac{m_1 + m_2}{2}\right)$	s
5	2.28±.065	.118±.092	.090±.080	.52(663df)

Compared to Type I results, these results favor pivoting and give less advantage to Cholesky, but the Type I, vs. Type II disparity is not very significant.

Forsythe and Moler [3, p. 50] present a heuristic argument to the effect that, on a machine with d decimal digits, a matrix of condition 10^c (the condition here is the ratio of the smallest and largest eigenvalues) should be inverted with (d - c) correct digits. It should be evident from the scatter plot, Figure 1, that condition alone is not a very good predictor. Only 2/3, or 68%, of the variance in correct digits is accounted for by the linear prediction by log condition.

Several alternatives to condition were tried to see how well they predict digits of inversion accuracy. These included Tmax (log of the smallest tolerance), Tr (log trace of the inverse correlation matrix), M (log maximum diagonal element in the inverse correlation

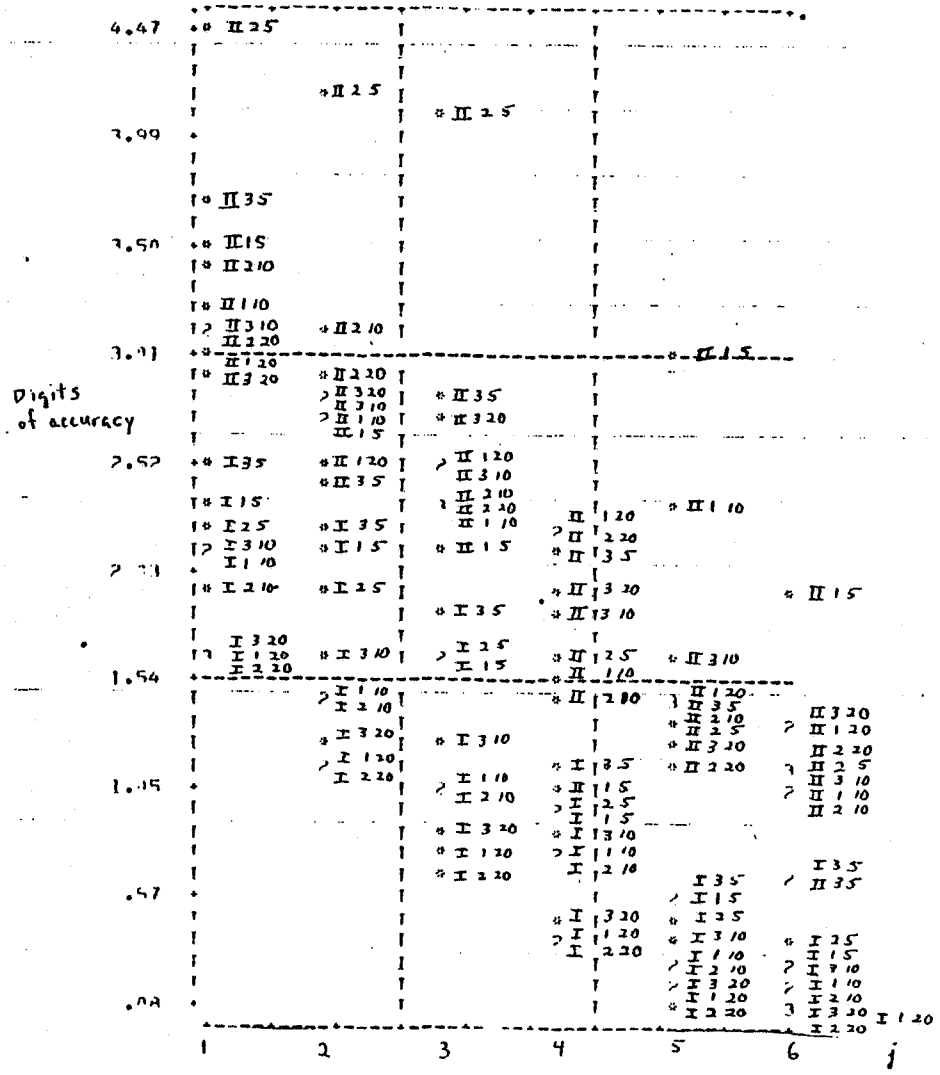


FIGURE 1 DIGITS OF ACCURACY VS j

(the smallest eigenvalue is $10^{-(j+7)/2}$)

Type : I, II Method : 1 = G-J, 2 = G-J-Piv, 3 = Chol Size : 5, 10, 20

TYPE	N	METH	J	DIGITS	TYPE	N	METH	J	DIGITS
1.	5.	1.	1.	2.39	2.	5.	1.	1.	3.54
1.	5.	2.	1.	2.32	2.	5.	2.	1.	4.47
1.	5.	3.	1.	2.61	2.	5.	3.	1.	3.77
1.	5.	1.	2.	2.21	2.	5.	1.	2.	2.74
1.	5.	2.	2.	1.98	2.	5.	2.	2.	4.18
1.	5.	3.	2.	2.26	2.	5.	3.	2.	2.52
1.	5.	1.	3.	1.67	2.	5.	1.	3.	2.22
1.	5.	2.	3.	1.73	2.	5.	2.	3.	4.12
1.	5.	3.	3.	1.90	2.	5.	3.	3.	2.82
1.	5.	1.	4.	1.02	2.	5.	1.	4.	1.14
1.	5.	2.	4.	1.04	2.	5.	2.	4.	1.69
1.	5.	3.	4.	1.10	2.	5.	3.	4.	2.20
1.	5.	1.	5.	.59	2.	5.	1.	5.	3.03
1.	5.	2.	5.	.51	2.	5.	2.	5.	1.49
1.	5.	3.	5.	.64	2.	5.	3.	5.	1.46
1.	5.	1.	6.	.36	2.	5.	1.	6.	2.02
1.	5.	2.	6.	.45	2.	5.	2.	6.	1.24
1.	5.	3.	6.	.75	2.	5.	3.	6.	.73
1.	10.	1.	1.	2.13	2.	10.	1.	1.	3.27
1.	10.	2.	1.	2.03	2.	10.	2.	1.	3.41
1.	10.	3.	1.	2.22	2.	10.	3.	1.	3.20
1.	10.	1.	2.	1.54	2.	10.	1.	2.	2.73
1.	10.	2.	2.	1.49	2.	10.	2.	2.	3.14
1.	10.	3.	2.	1.66	2.	10.	3.	2.	2.84
1.	10.	1.	3.	1.13	2.	10.	1.	3.	2.36
1.	10.	2.	3.	1.11	2.	10.	2.	3.	2.41
1.	10.	3.	3.	1.28	2.	10.	3.	3.	2.55
1.	10.	1.	4.	.81	2.	10.	1.	4.	1.55
1.	10.	2.	4.	.77	2.	10.	2.	4.	1.53
1.	10.	3.	4.	.87	2.	10.	3.	4.	1.88
1.	10.	1.	5.	.34	2.	10.	1.	5.	2.33
1.	10.	2.	5.	.29	2.	10.	2.	5.	1.45
1.	10.	3.	5.	.42	2.	10.	3.	5.	1.73
1.	10.	1.	6.	.26	2.	10.	1.	6.	1.13
1.	10.	2.	6.	.19	2.	10.	2.	6.	1.13
1.	10.	3.	6.	.35	2.	10.	3.	6.	1.17
1.	20.	1.	1.	1.69	2.	20.	1.	1.	3.04
1.	20.	2.	1.	1.69	2.	20.	2.	1.	3.12
1.	20.	3.	1.	1.72	2.	20.	3.	1.	3.00
1.	20.	1.	2.	1.22	2.	20.	1.	2.	2.56
1.	20.	2.	2.	1.21	2.	20.	2.	2.	2.49
1.	20.	3.	2.	1.29	2.	20.	3.	2.	2.87
1.	20.	1.	3.	.86	2.	20.	1.	3.	2.56
1.	20.	2.	3.	.71	2.	20.	2.	3.	2.38
1.	20.	3.	3.	.93	2.	20.	3.	3.	2.73
1.	20.	1.	4.	.45	2.	20.	1.	4.	2.24
1.	20.	2.	4.	.39	2.	20.	2.	4.	2.23
1.	20.	3.	4.	.49	2.	20.	3.	4.	1.94
1.	20.	1.	5.	.19	2.	20.	1.	5.	1.53
1.	20.	2.	5.	.14	2.	20.	2.	5.	1.22
1.	20.	3.	5.	.21	2.	20.	3.	5.	1.28
1.	20.	1.	6.	.08	2.	20.	1.	6.	1.35
1.	20.	2.	6.	.08	2.	20.	2.	6.	1.24
1.	20.	3.	6.	.09	2.	20.	3.	6.	1.39

TABLE 3 SIMULATION DATA

matrix), and C (log condition of the correlation matrix). Let J be the log of the condition of the matrix. Then the simple correlations with the number of correct digits of inversion are

Tmax	Tr	M	C	J
-.60	-.77	-.71	-.66	-.68.

This suggests the extremely surprising result that the trace of the inverse correlation matrix may be better than condition as an indication of the number of correct digits. More work is needed to verify this. Note that Tr, M, and C are closely related [1].

3. Summary

To summarize the results of the study, the Type I - Type II disparity is the most striking result. The difference in digits of accuracy between the two types is much greater than any of the observed Choleski vs. Gauss-Jordan differences. The author is unaware of any published analysis that would predict such a disparity. The much greater variability of the Type II results also requires explanation.

The differences favoring Choleski were expected. They do not seem sufficiently great to cause Gauss-Jordan users to switch, especially since the Choleski advantage is below one-tenth of a digit for large Type I matrices.

The condition of the matrix was a disappointing predictor of the number of correct digits. Surprisingly, the trace of the inverse correlation matrix did somewhat better.

References

[1] Berk, K. N. (1977) Tolerance and Condition in Regression Computations. Journal of the American Statistical Association,72, 863-866.

[2] Efroymson, M. A. (1960) Multiple Regression Analysis. In Mathematical Methods for Digital Computers. A. Ralston and H. S. Wilf, Eds, Wiley, New York.

[3] Forsythe, G. and Moler, C. B. (1967) Computer Solution of Linear Algebraic Equations, Prentice-Hall, Englewood Cliffs, N. J.

[4] Lewis, T. G. and Payne, W. H. (1973) Generalized feedback shift register psendorandom number algorithm. Journal of the Association for Computing Machinery, 20, 456-468.

[5] Marsaglia, G. (1962) Random variables and computers. Transactions of the Third Prague Conference. 499 - 512.

[6] Stewart, G. W. (1973) Introduction to Matrix Computations, Academic Press, New York.

[7] Wilkinson, J. H. (1961) Error analysis of direct methods of matrix inversion. J. Ass. Comp. Mach. 9, 281 - 330.

NUMERICAL SOLUTIONS OF THE BETA DISTRIBUTION

Hubert Bouver, APL of Johns Hopkins
Rolf E. Bargmann, The University of Georgia

ABSTRACT

This paper presents the derivation of a newly developed formula, with its computational algorithm, and its FORTRAN IV module for comparison of numerical methods in the evaluation of the Beta distribution.

Series and continued fraction expansions were compared with the goal of finding the most efficient techniques for the different domain of the two degrees of freedom of the Beta distribution. These methods, in addition to the standard serie solutions and continued fraction expansions, include the recent technique of the Hermite expansion around a local maximum which was investigated for large values of the two shape parameters of the Beta distribution.

The Incomplete Beta subprogram function was written in standard FORTRAN to evaluate in double precision the cumulative density function, the inverse of the cumulative density function and, the probability density function of the Beta distribution with guaranteed precision of 10 significant digits.

INTRODUCTION

A comparison of modern computational algorithms, for mathematical functions (e.g. IBM library [1]), with those used twenty years ago, shows a trend toward higher efficiency with guaranteed precision. Even for elementary trigonometric, exponential and logarithmic functions, the classical series expansions have been replaced by optimized fixed-length continued fractions and Chebyshef minimax rational functions.

The collection of mathematical functions by Abramowitz and Stegun [2] have been used extensively, especially the formulas and mathematical properties of series expansions and rational fractions. Johnson and Kotz [3], describe in detail properties of many statistical distribution functions and present formulas especially developed for approximations. They devote particular attention to formulas for small range of arguments and for modest precision.

The techniques of numerical analysis are, for the most part, well known and are merely studied as they relate to statistical distribution functions. However, the Hermite expansion around a maximum, as described next, appears to be a novel approach. It seems to have superficial similarity with a method described by Daniel [4] which Kendall and Stuart [5] regarded as an entirely novel approach for the evaluation of distributions. The Hermite expansion proved very successful for large values of the shape parameter ($\alpha > 100$) of the Incomplete Gamma and was needed to fill rather large gaps between continued fractions and Normal approximations Bouver and Bargmann [6].

If one is satisfied with low precision (e.g. 3 places) and a limited range of probabilities (e.g. 0.01 and 0.99 level) reference to the central limit theorem, variance stabilization transformations, and other approximations may be quite adequate. On the other hand, if high precision is required, these stand-by approximations are useless.

Application of the Hermite expansion to the Beta distribution

Let $J_{m,n}$ be defined as an Incomplete Beta function

$$J_{m,n}(x) = \frac{\Gamma(m+n+2)}{\Gamma(m+1)\Gamma(n+1)} \int_0^x t^m(1-t)^n \, dt, \qquad (1)$$

where $J_{m,n}(x) = B(x;m+1,n+1)$, $0 < x < 1$ and $m > 0$, $n > 0$ (m and n not necessarily integers).

Substituing $f(t) = m\log(t) + n\log(1-t)$ in (1) we obtain

$$J_{m,n}(x) = \frac{\Gamma(m+n+2)}{\Gamma(m+1)\Gamma(n+1)} \int_0^x e^{f(t)} \, dt, \qquad (2)$$

and its r^{th} derivative is

$$f^{(r)}(t) = (r-1)!(m+n)\left[\frac{(-1)^{r-1}}{p^{r-1}} - \frac{1}{q^{r-1}}\right].$$

Since

$f'(t) = \frac{m}{t} - \frac{n}{1-t}$, where $f'(\hat{t}) = 0$ implies $\hat{t} = \frac{m}{n+m} = p$

and

$f''(\xi) = -\frac{m}{\xi^2} - \frac{n}{(1-\xi)^2} < 0$ implies that $\hat{t}=p$ gives a local maximum, hence the Taylor series expansion around its local maximum p is

$$f(t) = (m+n)\left[p\log p + q\log q - \frac{1}{2pq}(t-p)^2 + \frac{1}{3}\left(\frac{1}{p^2} - \frac{1}{q^2}\right)(t-p)^3 - \frac{1}{4}\left(\frac{1}{p^3} + \frac{1}{q^3}\right)(t-p)^4 + \frac{1}{5}\left(\frac{1}{p^4} - \frac{1}{q^4}\right)(t-p)^5 - \ldots\right].$$

Evaluation of the Incomplete Beta Function

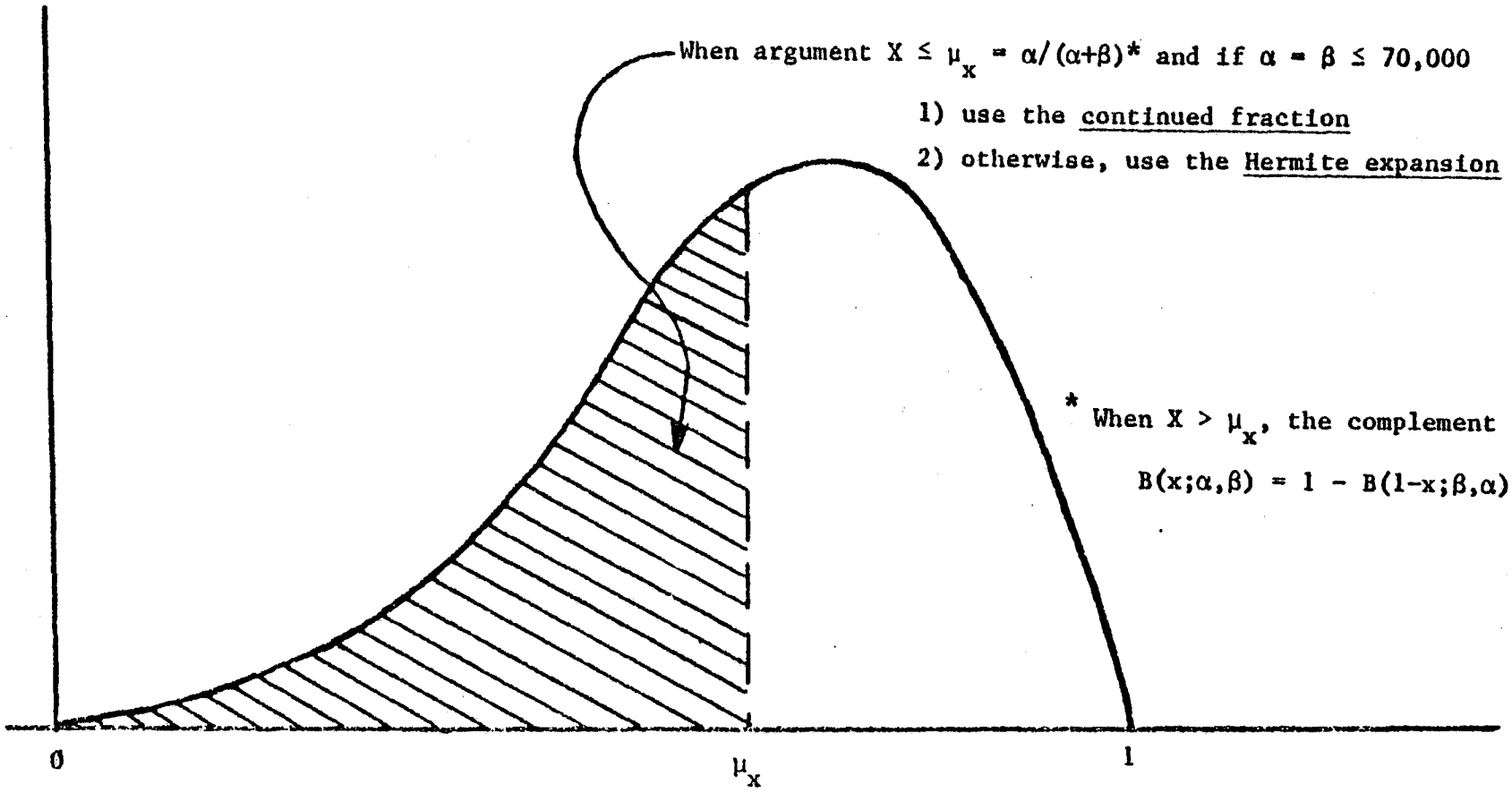


FIGURE 1

Thus (2) may be written as

$$J_{m,n}(x)=\frac{\Gamma(m+n+2)}{\Gamma(m+1)\Gamma(n+1)}\,p^m q^n \int_0^x e^{-\frac{1}{2pq}(t-p)^2(m+n)}\,e^{\hat{R}(t)}dt, \tag{3}$$

where

$$e^{\hat{R}(t)}=(m+n)\left[\frac{1}{3}\left(\frac{q^2-p^2}{(pq)^2}\right)(t-p)^3-\frac{1}{4}\left(\frac{q^3+p^3}{(pq)^3}\right)(t-p)^4\right.$$
$$\left.+\frac{1}{5}\left(\frac{q^4-p^4}{(pq)^4}\right)(t-p)^5-\ldots\right].$$

Let $z=\frac{(t-p)\sqrt{m+n}}{\sqrt{pq}}$ in (3). Then

$$J_{m,n}(x)=\frac{\Gamma(m+n+2)}{\Gamma(m+1)\Gamma(n+1)}\left(\frac{\sqrt{pq}\;p^m q^n\sqrt{2\pi}}{\sqrt{m+n}}\right)\int_{x_0}^{x_1}\phi(z)e^{R(z)}dz, \tag{4}$$

where $x_1=(x-p)\frac{\sqrt{m+n}}{\sqrt{pq}}$, $x_0=-\sqrt{\frac{m}{n}}\sqrt{m+n}$, $R(z)=\hat{R}(z\sqrt{\frac{pq}{m+n}}+p)$, and

$$e^{R(z)}=\frac{\alpha_1}{3}z^3-\frac{\alpha_2}{4}z^4+\frac{\alpha_3}{5}z^5-\frac{\alpha_4}{6}z^6+\frac{\alpha_5}{7}z^7-\ldots,$$

with

$$\alpha_1=\frac{q^2-p^2}{\sqrt{(m+n)pq}},\quad \alpha_2=\frac{q^3+p^3}{(m+n)pq},\quad \alpha_3=\frac{q^4-p^4}{[(m+n)pq]^{3/2}},$$

$$\alpha_4=\frac{q^5+p^5}{[(m+n)pq]^2},\quad \alpha_5=\frac{q^6-p^6}{[(m+n)pq]^{5/2}},\ \ldots.$$

Now, in the expansion of $e^{R(z)}$, it follows that

$$e^{R(z)}=\sum_{k=0}^{\infty}\frac{z^{3k}}{k!}\left[\sum_{j=0}^{\infty}(-1)^j\frac{\alpha_{j+1}}{j+3}z^j\right]^k,$$

and its final asymptotic expansion form up to terms involving $|(m+n)pq|^{-5}$ may be written as

$$e^{R(z)}=1+z^3\left(\frac{\alpha_1}{3}\right)-\left[z^4\frac{\alpha_2}{4}-z^6\frac{\alpha_1^2}{18}\right]+\left[z^5\frac{\alpha_3}{5}-z^7\frac{\alpha_1\alpha_2}{12}+z^9\frac{\alpha_1^3}{162}\right]$$
$$-\left[z^6\frac{\alpha_4}{6}-z^8\left(\frac{\alpha_1\alpha_3}{15}+\frac{\alpha_2^2}{32}\right)+z^{10}\frac{\alpha_1^2\alpha_2}{72}-z^{12}\frac{\alpha_1^4}{1944}\right]$$
$$+\left[z^7\frac{\alpha_5}{7}-z^9\left(\frac{\alpha_1\alpha_4}{18}+\frac{\alpha_2\alpha_3}{20}\right)+z^{11}\left(\frac{\alpha_1\alpha_2^2}{96}+\frac{\alpha_1^2\alpha_3}{90}\right)-z^{13}\frac{\alpha_1^3\alpha_2}{648}\right.$$
$$\left.+z^{15}\frac{\alpha_1^5}{29,160}\right]-\left[z^8\frac{\alpha_6}{8}-z^{10}\left(\frac{\alpha_1\alpha_5}{21}+\frac{\alpha_2\alpha_4}{24}+\frac{\alpha_3^2}{50}\right)\right.$$
$$+z^{12}\left(\frac{\alpha_1\alpha_2\alpha_3}{60}+\frac{\alpha_1^2\alpha_4}{108}+\frac{\alpha_2^3}{384}\right)-z^{14}\left(\frac{\alpha_1^2\alpha_2^2}{576}+\frac{\alpha_1^3\alpha_3}{810}\right)+z^{16}\frac{\alpha_1^4\alpha^2}{7,776}$$
$$\left.-z^{18}\frac{\alpha_1^6}{524,880}\right]+\left[z^9\frac{\alpha_7}{9}-z^{11}\left(\frac{\alpha_1\alpha_6}{24}+\frac{\alpha_2\alpha_5}{28}+\frac{\alpha_3\alpha_4}{30}\right)\right.$$
$$+z^{13}\left(\frac{\alpha_1\alpha_2\alpha_4}{72}+\frac{\alpha_1\alpha_3^2}{150}+\frac{\alpha_1^2\alpha_5}{126}+\frac{\alpha_2^2\alpha_3}{160}\right)-z^{15}\left(\frac{\alpha_1\alpha_2^3}{1,152}\right.$$
$$\left.+\frac{\alpha_1^2\alpha_2\alpha_3}{360}+\frac{\alpha_1^3\alpha_4}{972}\right)+z^{17}\left(\frac{\alpha_1^3\alpha_2^2}{5,184}+\frac{\alpha_1^4\alpha_3}{9,720}\right)-z^{19}\frac{\alpha_1^5\alpha^2}{116,640}$$
$$\left.+\frac{\alpha_1^7}{11,022,480}\right]-\left[z^{10}\frac{\alpha_8}{10}-z^{12}\left(\frac{\alpha_1\alpha_7}{27}+\frac{\alpha_2\alpha_6}{32}+\frac{\alpha_3\alpha_5}{35}\right)\right.$$
$$\left.+\frac{\alpha_4^2}{72}\right)+z^{14}\left(\frac{\alpha_1\alpha_2\alpha_5}{84}+\frac{\alpha_1\alpha_3\alpha_4}{90}+\frac{\alpha_1^2\alpha^6}{144}+\frac{\alpha_2\alpha_3^2}{200}+\frac{\alpha_2^2\alpha_4}{192}\right)$$
$$-z^{16}\left(\frac{\alpha_1\alpha_2^2\alpha_3}{480}+\frac{\alpha_1^2\alpha_2\alpha_4}{432}+\frac{\alpha_1^2\alpha_3^2}{900}+\frac{\alpha_1^3\alpha^5}{1,134}+\frac{\alpha_2^4}{6,144}\right)$$
$$+z^{18}\left(\frac{\alpha_1^2\alpha_2^3}{6,912}+\frac{\alpha_1^3\alpha_2\alpha_3}{3,240}+\frac{\alpha_1^4\alpha^4}{11,664}\right)-z^{20}\left(\frac{\alpha_1^4\alpha_2^2}{62,208}+\frac{\alpha_1^5\alpha^3}{145,800}\right)$$
$$\left.+z^{22}\frac{\alpha_1^6\alpha_2}{2,099,520}-z^{24}\frac{\alpha_1^8}{264,539,520}\right]+\left[z^{11}\frac{\alpha^9}{11}\right.$$
$$-z^{13}\left(\frac{\alpha_1\alpha_8}{30}+\frac{\alpha_2\alpha_7}{36}+\frac{\alpha_3\alpha_6}{40}+\frac{\alpha_4\alpha_5}{42}\right)+z^{15}\left(\frac{\alpha_1^2\alpha^7}{162}+\frac{\alpha_1\alpha_2\alpha_6}{96}\right.$$
$$\left.+\frac{\alpha_1\alpha_3\alpha_5}{105}+\frac{\alpha_1\alpha_4^2}{216}+\frac{\alpha_2^2\alpha^5}{224}+\frac{\alpha_2\alpha_3\alpha_4}{120}+\frac{\alpha_3^3}{750}\right) \tag{5}$$
$$-z^{17}\left(\frac{\alpha_1\alpha_2^2\alpha_4}{576}+\frac{\alpha_1\alpha_2\alpha_3^2}{600}+\frac{\alpha_1^2\alpha_2\alpha_5}{504}+\frac{\alpha_1^2\alpha_3\alpha_4}{540}+\frac{\alpha_1^3\alpha^6}{1,296}\right.$$
$$\left.+\frac{\alpha_2^3\alpha_3}{1,920}\right)+z^{19}\left(\frac{\alpha_1\alpha_2^4}{18,432}+\frac{\alpha_1^2\alpha_2^2\alpha_3}{2,880}+\frac{\alpha_1^3\alpha_2\alpha_4}{3,888}+\frac{\alpha_1^3\alpha_3^2}{8,100}\right.$$
$$\left.+\frac{\alpha_1^4\alpha^5}{13,608}\right)-z^{21}\left(\frac{\alpha_1^3\alpha_2^3}{62,208}+\frac{\alpha_1^4\alpha_2\alpha_3}{38,880}+\frac{\alpha_1^5\alpha_4}{174,960}\right)$$
$$+z^{23}\left(\frac{\alpha_1^5\alpha_2^2}{933,120}+\frac{\alpha_1^6\alpha_3}{2,624,400}\right)-z^{25}\left(\frac{\alpha_1^7\alpha_2}{44,089,920}\right)$$
$$\left.+z^{27}\left(\frac{\alpha_1^9}{7,142,567,040}\right)\right]-\left[z^{12}\frac{\alpha^{10}}{12}-z^{14}\left(\frac{\alpha_1\alpha_9}{33}\right.\right.$$
$$\left.+\frac{\alpha_2\alpha_8}{40}+\frac{\alpha_3\alpha_7}{45}+\frac{\alpha_4\alpha_6}{48}+\frac{\alpha_5^2}{98}\right)+z^{16}\left(\frac{\alpha_1^2\alpha_8}{180}+\frac{\alpha_1\alpha_2\alpha_7}{108}+\frac{\alpha_1\alpha_3\alpha_6}{120}\right.$$
$$\left.+\frac{\alpha_1\alpha_4\alpha_5}{126}+\frac{\alpha_2^2\alpha_6}{256}+\frac{\alpha_2\alpha_3\alpha_5}{140}+\frac{\alpha_2\alpha_4^2}{288}+\frac{\alpha_3^2\alpha_4}{300}\right)-z^{18}\left(\frac{\alpha_1\alpha_2^2\alpha_5}{672}\right.$$
$$\left.+\frac{\alpha_1\alpha_2\alpha_3\alpha_4}{360}+\frac{\alpha_1\alpha_3^3}{2,250}+\frac{\alpha_2^2\alpha_2\alpha_5}{576}+\frac{\alpha_2^2\alpha_3\alpha_5}{630}+\frac{\alpha_2^2\alpha_4^2}{1,296}+\frac{\alpha_2^3\alpha_7}{1,458}\right.$$
$$\left.+\frac{\alpha_2^3\alpha_3^2}{1,600}+\frac{\alpha_2^3\alpha_4}{2,304}\right)+z^{20}\left(\frac{\alpha_1\alpha_2^3\alpha_3}{5,760}+\frac{\alpha_1^2\alpha_2^2\alpha_4}{3,456}+\frac{\alpha_1^2\alpha_2\alpha_3^2}{3,600}\right.$$
$$\left.+\frac{\alpha_1^3\alpha_2\alpha_5}{4,536}+\frac{\alpha_1^3\alpha_3\alpha_4}{4,860}+\frac{\alpha_1^4\alpha_6}{15,552}+\frac{\alpha_2^5}{122,880}\right)-z^{22}\left(\frac{\alpha_1^2\alpha_2^4}{110,592}\right.$$
$$\left.+\frac{\alpha_1^3\alpha_2^2\alpha_3}{25,920}+\frac{\alpha_1^4\alpha_2\alpha_4}{46,656}+\frac{\alpha_1^4\alpha_3^2}{97,200}+\frac{\alpha_1^5\alpha_5}{204,120}\right)+z^{24}\left(\frac{\alpha_1^4\alpha_2^3}{746,496}\right.$$
$$\left.+\frac{\alpha_1^5\alpha_2\alpha_3}{583,200}+\frac{\alpha_1^6\alpha_4}{3,149,280}\right)-z^{26}\left(\frac{\alpha_1^6\alpha_2^2}{16,796,160}\right.$$
$$\left.+\frac{\alpha_1^7\alpha_3}{55,112,400}\right)+z^{28}\left(\frac{\alpha_1^8\alpha_2}{1,058,158,080}\right.$$
$$\left.-z^{30}\left(\frac{\alpha_1^{10}}{214,277,011,200}\right)\right]$$

The computer program DBETAX evaluates the Incomplete Beta function (4) in double precision. It is most valuable if both parameter are large. For $\alpha = \beta = 1000$, its precision is about 23 significant digits at the mean and 18 significant digits at $\mu \pm 5\sigma$. If one parameter is small (e.g. $\alpha = 1000$ and $\beta = 10$), the precision is about 10 significant digits at the mean and 8 significant digits toward either end.

Application of the continued fraction to the Beta distribution

We will show that the continued fraction is by far the most efficient method to evaluate the Incomplete Beta function. Its continued fraction may be written as [2]

$$B(x;\alpha,\beta)=\frac{x^\alpha(1-x)^\beta}{\Gamma(\alpha+1)\Gamma(\beta)}\left[\frac{1}{1+}\;\frac{c_1}{1+}\;\frac{c_2}{1+}\;\frac{c_3}{1+}\cdots\frac{c_n}{1+}\cdots\right]$$

where

$$c_{2k}=\frac{-(\alpha+k-1)(\alpha+\beta+k-1)}{(\alpha+2k-2)(\alpha+2k-1)}x,$$

and

$$c_{2k+1}=\frac{k(\beta-k)}{(\alpha+2k-1)(\alpha+2k)}x,\quad k=1,2,3\ldots$$

This formula is most efficiently used when x is less than its mean (i.e. $x < \mu_x = \alpha/(\alpha+\beta)$), but when $x > \mu_x$, we may simply take complements (i.e. $B(x;\alpha,\beta) = 1 - \beta(1-x;\beta,\alpha)$).
The computer program DBETAX evaluates the Incomplete Beta function using the continued fraction as defined in the following way [2]:

$$f_n=\frac{A_n}{B_n}=\frac{a_1}{b_1+\dfrac{a_2}{b_2+\dfrac{a_3}{b_3+a_4}}}=\frac{a_1}{b_1+}\;\frac{a_2}{b_2+}\;\frac{a_3}{b_3+}\cdots\frac{a_n}{b_n}$$

FLOW CHART FOR THE

INCOMPLETE BETA FUNCTION

BETAX

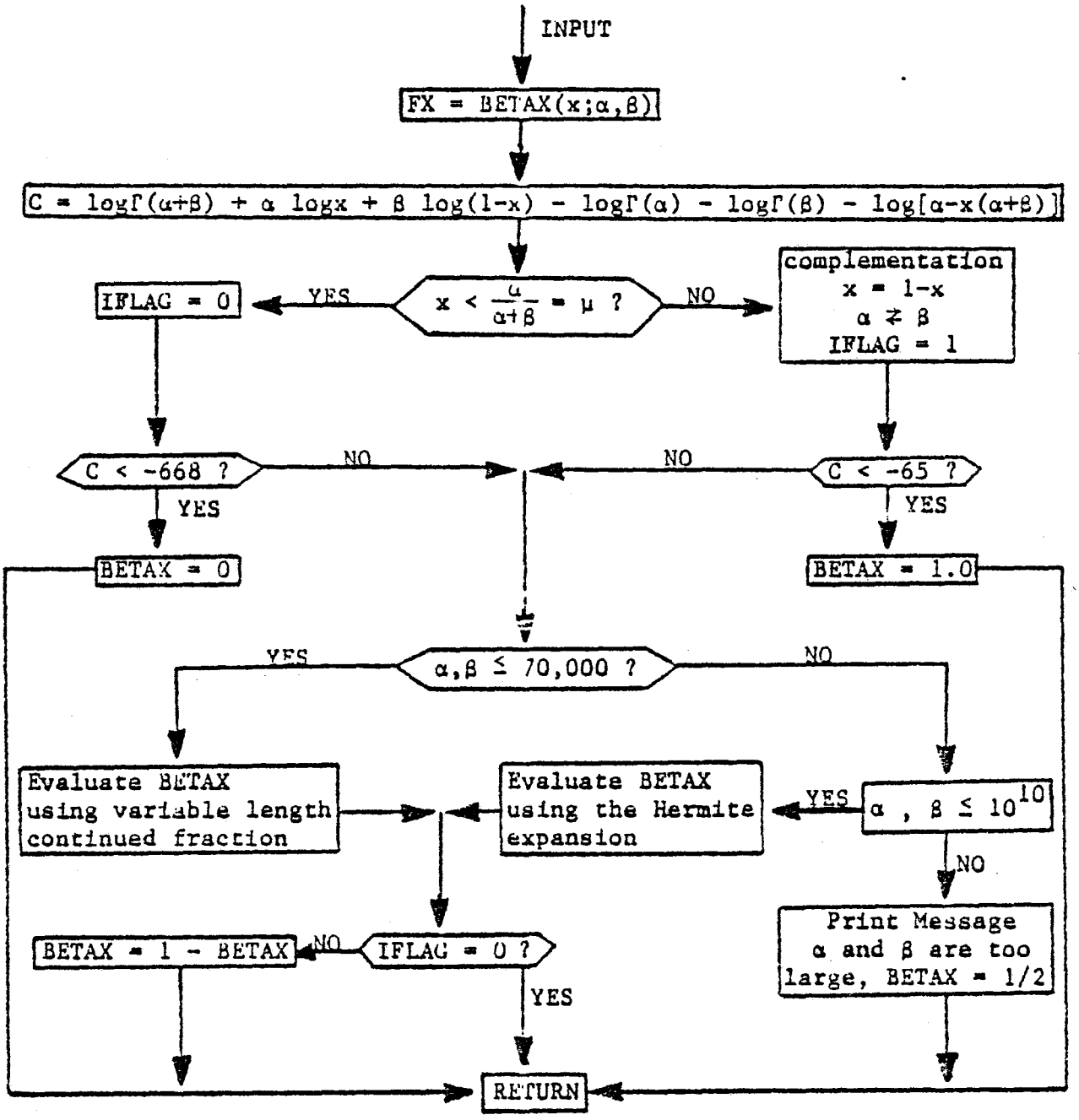


FIGURE 2

Time Comparison on an Average of 10 Calls in msec, for 10 Digits Precision, using Variable Length Continued Fraction and Hermite Expansion for the Evaluation of the Incomplete Beta Function *

Parameters $\alpha = \beta$	Continued Fraction				Hermite Expansion	
	Time For X		Cycle For X		Time For X	
	$\mu_x - 3\sigma_x$	μ_x	$\mu_x - 3\sigma_x$	μ_x	$\mu_x - 3\sigma_x$	μ_x
10	2.2	2.8	6	9	39.0	37.4
100	3.5	3.8	11	20	37.7	35.0
1,000	3.7.	7.9	16	43	36.9	33.5
10,000	3.9	17.5	21	91	36.4	33.5
40,000	5.8	26.0	23	144	37.1	34.0
70,000	5.8	30.8	23	174	35.3	34.1
100,000					34.8	34.1
10^8					35.0	34.2
10^{10}					35.2	34.1

TABLE 1

* if $x > \mu_x = \alpha/(\alpha+\beta)$, the complement (i.e. $B(x;\alpha,\beta) = 1 - B(1-x;\beta,\alpha)$

where

$$A_n = b_n A_{n-1} + a_n A_{n-2} \quad , \quad B_n = b_n B_{n-1} + a_n B_{n-2} \quad .$$

Our computer program DBETAX uses this Euler second-difference, two-step per cycle continued fraction method as defined by the following FORTRAN-type statement where for each cycle (k = 1,2,3, ...) we have

$$AEV = a_{2k}, \quad AOD = a_{2k+1}, \quad BEV = b_{2k}, \quad BOD = b_{2k+1},$$

$$\text{cycle}\begin{cases}\text{step 1}\begin{cases}ALO=BEV*AHI+AEV*ALO \rightarrow A_2,A_4,A_6, \cdots A_{n-1}\\ BLO=BEV*BHI+AEV*BLO \rightarrow B_2,B_4,B_6, \cdots B_{n-1}\end{cases}\\ \text{step 2}\begin{cases}AHI=BOD*ALO+AOD*AHI \rightarrow A_3,A_5,A_7, \cdots A_n\\ BHI=BOD*BLO+AOD*BHI \rightarrow B_3,B_5,B_7, \cdots B_n\end{cases}\end{cases}.$$

Initially

$$ALO = A_0 = 0, \qquad AHI = A_1 = a_1$$

$$BLO = B_0 = 1, \qquad BHI = B_1 = b_1$$

and the result is A_n/B_n

$$F = AHI \;/\; BHI$$

Application of the series expansion to the Beta distribution

The Incomplete Beta function as previously defined, may be written as

$$B(x;\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \int_0^x t^{\alpha-1}(1-t)^{\beta-1}\,dt$$

$$= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha+1)\Gamma(\beta)} x^{\alpha}(1-x)^{\beta} + B(x;\alpha+1,\beta)$$

which is a recurrence relation valid for all $\alpha > 0$, $\beta > 0$; thus we have an infinite series expansion

$$B(x;\alpha,\beta) = \frac{(1-x)^{\beta}}{\Gamma(\beta)}\left[\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)}x^{\alpha} + \frac{\Gamma(\alpha+\beta+1)}{\Gamma(\alpha+1)}x^{\alpha+1} + \frac{\Gamma(\alpha+\beta+2)}{\Gamma(\alpha+2)}x^{\alpha-2} + \ldots\right]$$

$$= \sum_{k=0}^{\infty} \binom{\alpha+\beta+k-1}{\alpha+k} x^{\alpha-k}(1-x)^{\beta} \quad .$$

This series (see, e.g. [2]) is the expansion in terms of "negative binomial" probabilities; the expansion in "binomial" terms is less useful for non-integer α and β since, in the latter, terms will have alternating signs. For high precision evaluation, a series expansion is not very efficient for the Incomplete Beta function, unless x is very small ($x < \mu_x - 5\sigma_x$, say). The series expansion does, however, play an important role in the evaluation of non-central Beta distributions since, by an adroit procedure of storing individual terms of the series, the non-central distributions may be evaluated in approximately the same amount of CPU time as the central one.

CONCLUSION

For the Incomplete Beta function, the continued fraction evaluation with a variable number of terms, for 10 digits of precision, has proven to be vastly superior for all cases encountered in practice. Therefore, the continued fraction is the only module considered for the evaluation of the Incomplete Beta function unless $\min(\alpha,\beta) \geq 70{,}000$. For this extreme region, the Hermite expansion will be used. When $\min(\alpha,\beta) = 10^8$, the results of the Hermite expansion are valid to 5 digits of precision; beyond $\min(\alpha,\beta) = 10^9$, the results are even less accurate. At this time, the highest value for which the Incomplete Beta function will be evaluated is at $\min(\alpha,\beta) = 10^{10}$ where the results are still valid to 3 places of precision. No approach resulting in reasonable precision has been found for values of $\min(\alpha,\beta) > 10^{10}$.

The continued fraction works very efficiently even if the Incomplete Beta function is extremely J-shaped or extremely U-shaped (e.g. $\alpha = 10^{-10}$ and $\beta = 10^{-8}$). The continued fraction becomes more time-consuming as the argument x approaches the mean (approach is always from the left for, if $x > \mu_x = \alpha/(\alpha+\beta)$, the complement is used, (i.e., $B(x;\alpha,\beta) = 1 - B(1-x;\beta,\alpha)$).

Table 1 and Figure 1 show for a few selected values, the average time of execution of DBETAX for 10 digits of precision. The flow-chart (see figure 2) and the program listing of DBETAX in its final form, Figures 3 and 4 represent the Statistical Distribution Package SDP10 and its calling sequence are illustrated.

REFERENCES

[1] IBM Systems Reference Library, Mathematical and Service Subprograms, IBM GC 28-6818, 1972.

[2] Abramowitz, M. and Stegun, I.A., Handbook of Mathematical Functions, National Bureau of Standards, Washington, D.C., 1968.

[3] Johnson, N. L. and Kotz, S., Continuous Univariate Distributions-1,2. Houghton-Mifflin Co., 1970.

[4] Daniels, H. E., 'Saddlepoint Approximations in Statistics'. Ann. Math. Statist., 25, 631, 1954.

[5] Kendall, M. G. and Stuart, A., The Advanced Theory of Statistics, Vol. I., London, Griffin, 1969.

[6] Bouver, H and Bargmann, R.E., Numerical Solutions of the Incomplete Gamma Function. Proceeding of the Tenth Interface Symposium on Computer Science and Statistics, 1977.

[7] Bouver, H. and Bargmann, R. E., The Pearsonian Distribution Package: An Application to Curve Fitting. Proceeding of the 1977 American Statistical Association of the Statistical Computing Section, 1977.

STATISTICAL DISTRIBUTION PACKAGE
(SDP10)

STATISTICAL DISTRIBUTION	PROGRAM † NAME	EVALUATION OF THE CUMULATIVE DISTRIBUTION FUNCTION AS AN EXAMPLE OF ITS USAGE
NORMAL	DYORM [X P Z]	$Y = \text{DYORMX}(X) \rightarrow \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{X} e^{-\frac{t^2}{2}}\, dt$
GAMMA	DGAM [X P Z]	$Y = \text{DGAMX}(X,\alpha) \rightarrow \frac{1}{\Gamma(\alpha)} \int_{0}^{X} t^{\alpha-1} e^{-t}\, dt$
CHI-SQUARE	DCHI [X P Z]	$Y = \text{DCHIX}(X,\alpha) \rightarrow \frac{1}{2^{\frac{\alpha}{2}}\,\Gamma(\frac{\alpha}{2})} \int_{0}^{X} t^{(\alpha-1)/2} e^{-t/2}\, dt$
T	DTT [X P Z]	$Y = \text{DTTX}(X,\alpha) \rightarrow \frac{\Gamma(\frac{\alpha+1}{2})}{\sqrt{\alpha\pi}\,\Gamma(\frac{\alpha}{2})} \int_{-\infty}^{X} \frac{1}{\left(1+\frac{t^2}{\alpha}\right)^{\frac{(\alpha+1)}{2}}}\, dt$
BETA	DBETA [X P Z]	$Y = \text{DBETAX}(X,\alpha,\beta) \rightarrow \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \int_{0}^{X} t^{\alpha-1} (1-t)^{\beta-1}\, dt$
F	DFF [X P Z]	$Y = \text{DFFX}(X,\alpha,\beta) \rightarrow \frac{\Gamma(\frac{\alpha+\beta}{2})}{\Gamma(\frac{\alpha}{2})\Gamma(\frac{\beta}{2})} (\frac{\alpha}{\beta})^{\frac{\alpha}{2}} \int_{0}^{X} \frac{t^{\frac{(\alpha-2)}{2}}}{\left(1+\frac{\alpha}{\beta}t\right)^{\frac{(\alpha+\beta)}{2}}}\, dt$
Log Gamma Log($\Gamma(\alpha)$)	DDLGGM	$R = \text{DDLGGM}(x) \rightarrow \int_{0}^{\infty} e^{-t} t^{\alpha-1}\, dt$

† X indicates the integral CDF
P indicates the percentage points (DYORMP is the inverse function of DYORMX)
Z indicates the ordinate (PDF)

STATISTICAL DISTRIBUTION PACKAGE CALLING SEQUENCE

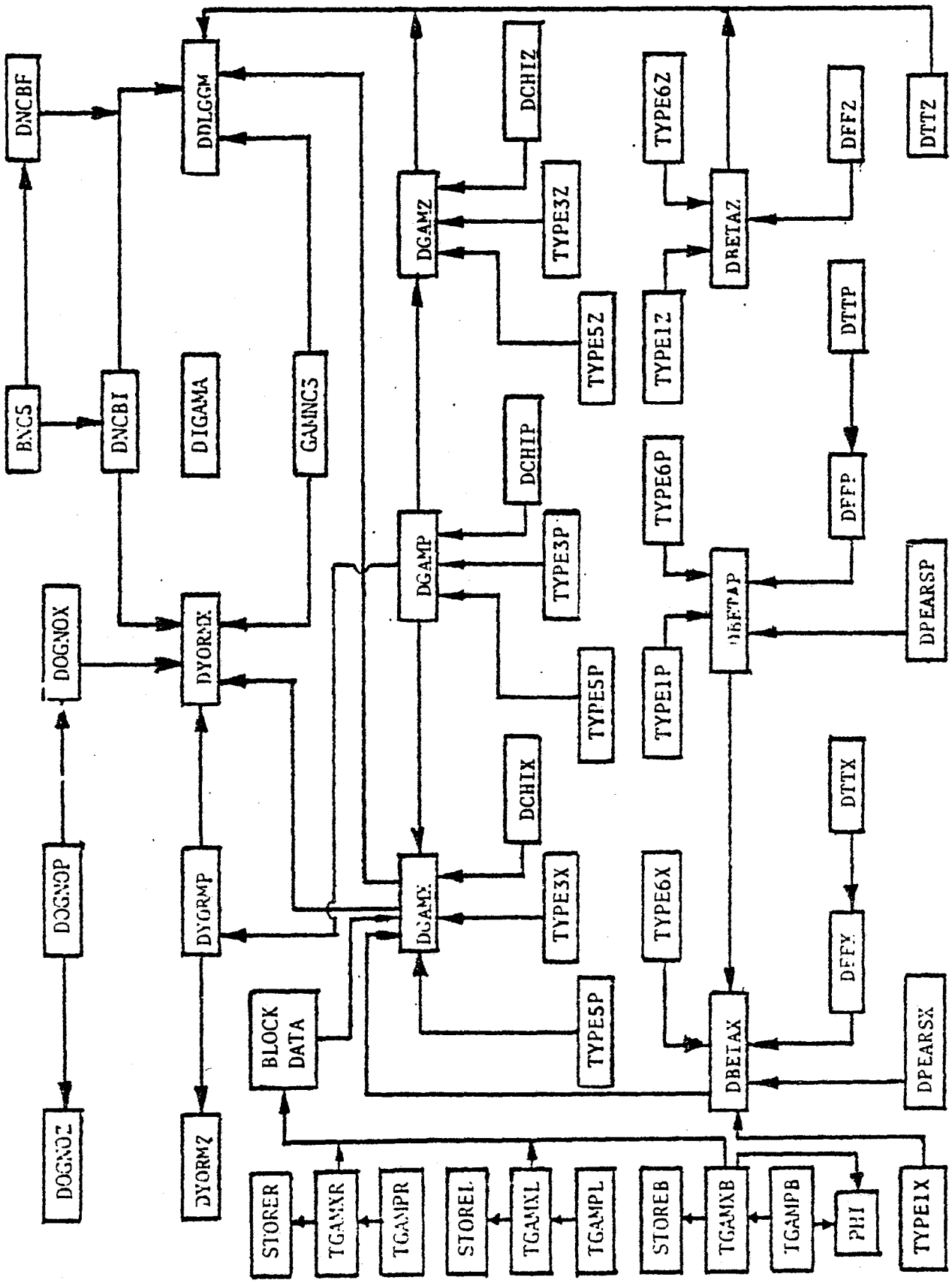


FIGURE 4

```
C ------------------------------ BETAX ------------------------------ C
C                                                                      C
C     WRITTEN BY   HUBERT BOUVER AND ROLF E. BARGMANN                  C
C                                                                      C
C                  DEPARTMENT OF STATISTICS AND COMPUTER SCIENCE       C
C                                                                      C
C                  UNIVERSITY OF GEORGIA. ATHENS GEORGIA.              C
C                                                                      C
C                                                                      C
C                                                                      C
C        THIS 6400 CDC SUBPROGRAM FUNCTION EVALUATES THE CUMULATIVE    C
C     DISTRIBUTION OF THE INCOMPLETE BETA FUNCTION.                    C
C                                                                      C
C                                                                      C
C       (1)  THE FUNCTION CALLING STATEMENT.                           C
C                                                                      C
C          P = BETAX (X.ALPHA.BETA)                                    C
C                                                                      C
C          WHERE  P = PROBABILITY LEVEL.                               C
C                                                                      C
C                 X = THE PERCENTAGE POINT.I.E THE UPPER LIMIT         C
C                                                                      C
C                       OF THE C.D.F.. 0 .GT. X .LT. 1.0               C
C                                                                      C
C                 ALPHA = THE FIRST SHAPE PARAMETER                    C
C                                                                      C
C                 BETA = THE SECOND SHAPE PARAMETER.                   C
C                                                                      C
C                                                                      C
C       (2)  THE PROGRAM LIMITATION                                    C
C                                                                      C
C            IF THE SUM OF THE TWO PARAMETERS EXCEED  1.E8  THE        C
C          RESULTED VALUE WILL BE SET TO  0.5.                         C
C                                                                      C
C                                                                      C
C       (3)  REFERENCES                                                C
C                                                                      C
C           ABRAMOWITZ. M AND STEGUM. I. HANDBOOK OF MATHEMATICAL      C
C                                                                      C
C        FUNCTIONS. NEW YORK. DOVER. 1964.                             C
C                                                                      C
C           BARGMANN. ROLF E.. A STATISTICAL DISTRIBUTION PACKAGE.     C
C                                                                      C
C        DEPARTMENT OF STATISTIC AND COMPUTER SCIENCES. UOA. ATHENS.   C
C                                                                      C
C                                                                      C
C ---------------------------------------------------------------------C
C
C
                    FUNCTION BETAX (X.ALPHA.BETA)
C
C *** ALL CONSTANTS USED IN THIS PROGRAM ARE IN DATA STATEMENT
```

```
C
      DIMENSION B1(30),C1(30)
      LOGICAL FLAG
      X=XX
      A=AA
      B=BB
      FLAG=.FALSE.
      IF(X.LT.0.0) GO TO 95
      IF(A.LT.0.0.OR.B.LT.0.0) GO TO 98
      BETAX=1.0
      IF(X.GT.1.0) GO TO 95
C
      C=DLGGM(A+B)+A*ALOG(X)+B*ALOG(1.0-X)
     $ -DLGGM(A)-DLGGM(B)-ALOG(A-X*(A+B))
      IF(X.LT.A/(A+B)) GO TO 10
      X=1.0-X
      A=BB
      B=AA
      FLAG=.TRUE.
      IF(C.LT.-34.0) GO TO 99
   10 IF(C.LT.-575.0) GO TO 96
      IF(A+B.LT.1.E5) GO TO 20
      IF(A+B.LT.1.E10) GO TO 30
      BETAX=0.5
      WRITE(6,102) A,B
  102 FORMAT(//,* THE PARAMETERS ARE TOO LARGE*,2G15.6)
      GO TO 99
   20 CONTINUE
      APB=A+B
      ALO=RM=0.0
      BLO=BHI=BEV=BOO=1.0
      AHI=EXP(DLGGM(APB)+A*ALOG(X)+B*ALOG(1.0-X)-
     $ DLGGM(A+1.0)-DLGGM(B))
      F=FX=AHI
   11 RM=RM+1.0
      RM1=RM-1.0
      AEV=-(A+RM1)*(A+B+RM1)*X/((A+RM1+RM1)*(A+RM+RM1))
      AOO=RM*(B-RM)*X/((A+RM+RM1)*(A+RM+RM))
      ALO=BEV*AHI+AEV*ALO
      BLO=BEV*BHI+AEV*BLO
      AHI=BOO*ALO+AOO*AHI
      BHI=BOO*BLO+AOO*BHI
      IF(BHI.EQ.0.0) GO TO 11
      F=AHI/BHI
      IF(ABS((F-FX)/F).LT.1.E-12) GO TO 21
      FX=F
      GO TO 11
   21 BETAX=F
C
   25 IF(FLAG) GO TO 99
      BETAX=1.0-BETAX
      GO TO  99
   30 CONTINUE
      BH=X
```

```
      RM=A-1.0
      RN=B-1.0
      DATA R1 /1.0/
      RMN=RM+RN
      P=RM/RMN
      Q=R1-P
      RPQ=RMN*P*Q
      N = 24
      A1=(Q-P)/SQRT(RPQ)
      A2=(Q**3+P**3)/RPQ
      A3=(Q**4-P**4)/RPQ**1.5
      A4=(Q**5+P**5)/RPQ**2
      A5=(Q**6-P**6)/RPQ**2.5
      A6=(Q**7+P**7)/RPQ**3
      A7=(Q**8-P**8)/RPQ**3.5
      A8=(Q**9+P**9)/RPQ**4
      B1(1)=A1**8/264539520.0
      B1(2)=0.0
      B1(3)=-A1**6*A2/2099520.0
      B1(4)=A1**7/11022480.0
      B1(5)=A1**4*A2**2/62208.0+A1**5*A3/145800.0
      B1(6)=-A1**5*A2/116640.0
      B1(7)=A1**6/524880.0-(A1**2*A2**3/6912.0+A1**3*A2*A3/3240.0
     $ +A1**4*A4/11664.0)
      B1(8)=A1**3*A2**2/5184.0+A1**4*A3/9720.0
      B1(9)=-A1**4*A2/7776.0+A1*A2**2*A3/480.0
     $ +A1**2*A2*A4/432.0+A1**2*A3**2/900.0
     $ +A1**3*A5/1134.0+A2**4/6144.0
      B1(10)=A1**5/29160.0-(A1*A2**3/1152.0
     $ +A1**2*A2*A3/360.0+A1**3*A4/972.0)
      B1(11)=A1**2*A2**2/576.0+A1**3*A3/810.0
     $ -(A1*A2*A5/84.0+A1*A3*A4/90.0+A1**2*A6/144.0
     $ +A2*A3**2/200.0+A2**2*A4/192.0)
      B1(12)=-A1**3*A2/648.0+A1*A2*A4/72.0+A1*A3**2/150.0
     $ +A1**2*A5/126.0+A2**2*A3/160.0
      B1(13)=A1**4/1944.0-(A1*A2*A3/60.0+A1**2*A4/108.0
     $ +A2**3/384.0)+A1*A7/27.0+A2*A6/32.0
     $ +A3*A5/35.0+A4**2/72.0
      B1(14)=A1*A2**2/96.0+A1**2*A3/90.0-(A1*A6/24.0
     $ +A2*A5/28.0+A3*A4/30.0)
      B1(15)=-A1**2*A2/72.0+A1*A5/21.0+A2*A4/24.0
     $ +A3**2/50.0-A8/10.0
      B1(16)=A1**3/162.0-(A1*A4/18.0+A2*A3/20.0)+A7/9.0
      B1(17)=A1*A3/15.0+A2**2/32.0-A6/8.0
      B1(18)=-A1*A2/12.0+A5/7.0
      B1(19)=A1**2/18.0-A4/6.0
      B1(20)=A3/5.0
      B1(21) = -A2/4.0
      B1(22) = A1/3.0
      B1(23) = B1(24) = 0.0
      B1(25) = 1.0
      NP = N+1
      CALL HERPOL  (B1,C1,N,2)
      NM = N-1
```

```
      CALL HERPOL  (C1,B1,NM,1)
      CN1 = C1(NP)
      AL = -SQRT(RM*RMN/RN)
      BN = BH
      BN=(BN-P)*SQRT(RMN/(P*Q))
C
      DATA IN1 /1/,RNO /0.00/
      SJ =    + CN1* (YORMX(BH)-YORMX(AL))
      PA = (YORMZ(AL))
      PB = (YORMZ(BH))
      NN = N-IN1
      SJ1 = SJ2 = RNO
      DO 12 I=IN1,NN
      NR = N -I
      BZ = B1(I)
      SJ1 = SJ1 + BZ*AL**NR
      SJ2 = SJ2 + BZ*BH**NR
   12 CONTINUE
      SJ = SJ + PA*(SJ1+B1(N)) - PB*(SJ2+B1(N))
C
      SJJ=EXP((RM+.50)*ALOG(P)+(RN+.50)*ALOG(Q)
     $  +OLGGM(RMN+2.0)-OLGGM(RM+1.0)-OLGGM(RN+1.0))*
     2  SQRT(8.0*ATAN(1.0)/RMN)
      BETAX=SJJ*SJ
C
      GO TO 25
   95 WRITE(6,100) X
  100 FORMAT(//,* ILLEGAL INPUT VALUES FOR X=,G15.6)
      GO TO 99
   96 BETAX=0.0
      GO TO 99
   98 WRITE(6,101) A,B
  101 FORMAT(//,* ILLEGAL INPUT PARAMETERS FOR A,B=,2G15.6)
   99 RETURN
      END
```

An Algorithm to Derive Mnemonics for Computer Usage

John Brode, Jeffrey Stamen, and Robert Wallace
23 Berkeley St., Cambridge, MA.

Commands to computer systems make extensive use of mnemonics. To regularize the formation of these mnemonics, an algorithm has been devised that takes word strings as input. Thus the user of this system need know only the algorithm and the full name for a procedure in order to create the correct mnemonic.

As the online use of computers expands beyond the computer professional to the computer consumer, the problem of keyword name proliferation arises. This has already occurred in the world of statistical computing where there are a great many statistical techniques each of which is assigned an abbreviation as a keyword to minimize typing (and also to fit environments where keywords are restricted to only 6 or 8 characters).

These abbreviations, however, are arbitrary. They have to be memorized for each usage. Besides, which analysis of variance should be called ANOVA since there are a variety of techniques available?

To get around these encombrances, an abbreviation algorithm was developed at M.I.T. (1) The advantage of an algorithmic approach is that the user is absolved from memorizing abbreviations. Instead, each abbreviation can be recreated from the normal name for a statistical technique by the application of the algorithm.

The algorithm for abbreviating names is as follows:

The first letter and the next following consonant (if any) of the first word to which are added the first letter of each subsequent word in the name. (N.B. prepositions, conjunctions, definite and indefinte articles are passed over in scanning the subsequent words.)

As an example, "analysis of variance for complete layouts" becomes "anvcl". "an" comes from the first word. "of" is skipped as is "for" so the first letters of the remaining words make up "vcl".

Exceptions to the above algorithm have been adopted for two categories of names:

1) In order to avoid confusion and redundancy, short names are not abbreviated. Short is defined as any name composed of only one word which has four or fewer letters. E.G., "for", "with", "plot", are not abbreviated.

2) Function names, such as "log", "tan," that are already widely used in an abbreviated form are left untouched. For the most part, these abbreviations have become names in themselves and as such would not be abbreviated under the above algorithm, exception 1. The few, like "conig", that would be abbreviated normally have been left untouched as well so as to avoid undue confusion.

---

(1) This algorithm was developed by the authors as part of their work at the Cambridge Project, M.I.T., and is incorporated into the DATATRAN language developed there. This research was supported in part by the Advanced Research Projects Agency of the Dept. of Defense under Contract No. F30602-72-C-0001, which was monitored by the Rome Air Development Center.

STATISTICAL DISTANCE MEASURES AND TEST SITE SELECTION: SOME CONSIDERATIONS

Charles D. Cowan and Randall K. Spoeri
Bureau of the Census

ABSTRACT

The selection of test sites or a judgemental sample of test items is often influenced by political and administrative considerations. Yet the researcher would like to make this selection as objective as possible, and at the same time be guaranteed that the selections made are representative of the domain from which the test items are selected. Often the selection process takes into account demographic and economic statistics about an area, gathered in a census or estimated from survey data.

One way of approaching this problem is to use statistical distance measures. The measures described in this paper were used to calculate a distance for each site in a set under consideration from a standard specified by the researcher. However, when two or more test sites are being selected, this method may not yield the best combination of sites or items in regard to the distance of the combination from the standard. This paper examines conditions under which consideration of individual distances may be inappropriate. However, it is shown how the individual distances may still be used, while improving the selection process for combinations of test sites.

Introduction

In April of 1977, the Farmers Home Administration (FmHA) of the U.S. Department of Agriculture entered into an interagency agreement with the Bureau of the Census. One objective of this agreement was to assist FmHA in selecting six states, two in each of three regions, as sites in which to test their Unified Management Information System. The method of selection used and compared three measures of distance, each offering an objective and quantitative basis to determine the "distance" of a state from its region. These measures were a) Euclidean distance, b) Mahalanobis distance (Mahalanobis (1936)), and c) a nonparametric ranking procedure suggested by Kendall (1975). Each distance measure is designed to quantify similarity.

In this project, FmHA wanted assistance in determining which states were most similar to a region simultaneously considering a number of factors. Thirty two characteristics, or variables, to be used in the determination of similarity of a state to a region were identified by Census and FmHA. These included such factors as median family income, land area, and FmHA caseload. For each of three FmHA regions, consisting of 9,9, and 12 states, Census was asked to indicate two states which might represent the region for testing of the information system.

In general, the three measures used produced very similar results. All three measures identified the same state as having the minimum distance for all three regions, and all measures indicated the same state as having the second smallest distance in two of the three regions. These two states in each region were selected as most representative of their region. It might be, however, that this procedure gives a set of items which jointly does not have the smallest distance from the regional averages. The problem will be stated more precisely in the next section.

Selection of Combinations of Sites or Items

The distinction being drawn in this paper is the difference in the selection obtained when calculating distances for individual sites, contrasted with results obtained when combinations of sites are considered simultaneously. When n objects are being compared, from which q are to be selected, there are $\binom{n}{q}$ combinations to be reviewed. To determine which combination yields the minimum distance from the standard established by the researcher, it will be shown that consideration must be given to both the individual distances of items and the distances between those items. Computing algorithms are presented as part of the discussion.

Methodology - Assumptions

The development that follows makes some assumptions to simplify the presentation, with no intentional loss in the generality of the argument. It will be assumed that all distances are calculated from the mean(s) of the variables under review. Thus if there are n observations and p variables, each observation can be represented as a point in a p dimensional space with a centroid of the observations located at the point $(\bar{X}_1, \bar{X}_2, \ldots, \bar{X}_p)$, where $\bar{X}_j$ denotes the mean of the n observations for the jth variable. This assumption is used only to simplify the calculations; any centroid can be specified. It may be that the researcher would prefer to calculate distances from medians rather than means, or from a minimum

tolerance level that would represent the point where cost savings would be greatest given a certain desired level of quality.

Secondly, following the recommendations of several authors (Chen (1974), Gower (1966), Kendall (1975)), the observations will be standardized by subtracting the mean from each variable, and dividing each variable by its standard deviation. If some other standard than the mean were desired (see above), one could subtract those values instead; the purpose of the standardization is to move the centroid to the origin. Division by the standard deviation makes all observations unitless and yields observations of approximately the same order of magnitude. This is necessary since some variables might be represented in millions of dollars, whereas others might be tabulated in square feet. In place of the standard deviation, one might also wish to use the variable mean (if not zero), the range, or a skewness measure.

The final assumption is that all sites or items receive equal consideration and equal weight. For example, when calculating the mean income for a region, the mean is calculated as the unweighted average of the individual states' incomes. Later in the discussion it becomes necessary to average the values of two observations, and so each is given equal weight. This assumption is made to facilitate the geometric presentation, but weights could be used.

Definitions

Let X be an n x p matrix consisting of n observations on p variables. The entries in X can be standardized by subtracting the mean of each variable from the n observations on that variable, and dividing each observation by its standard deviation. To subtract the means, we define a matrix E of size n x n, with all entries in E equal to unity. Then,

$$X^* = X - \frac{1}{n} EX.$$

From X*, a matrix V (the variance - covariance matrix) is calculated as:

$$V = \frac{1}{n-1} (X^*)' X^*$$

The principal diagonal of this matrix is extracted, and the reciprocals of the square roots of the diagonal elements are inserted in the principal diagonal of another p x p matrix W, whose off diagonal elements are zero.

The diagonal elements of W are the reciprocals of the standard deviations. The matrix Z is then derived as

$$Z = X^*W.$$

The set of standardized observations is now displayed in matrix form by Z, an n x p matrix, where n can be greater than, less than, or equal to p. An observation or case, the i^{th} case say, is the set of values on the p variables which make up the i^{th} row of Z. For the remainder of this paper, the term "observation" will be used to denote one of the n rows of Z, and may also be viewed as a point or vector in a p dimensional space. Also, "p-space" will refer to the p dimensional space under study.

The Euclidean distance of observation i from the origin is the square root of:

$$d_{ii} = \sum_{j=1}^{p} (z_{ij} - 0)^2 = \sum_{j} z_{ij}^2, \text{ or}$$

$d_{ii} = \underline{z}_i \underline{z}_i'$, where $\underline{z}_i$ is the i^{th} row of Z, and

$D = ZZ'$, where Z is as described above.

The n principal diagonal elements of D, the d_{ii}, are the squared Euclidean distances of the n standardized observations measured from the origin of the p-space. The off diagonal elements, the d_{ik} $(i \neq k)$, are the scalar products between observations, and do not directly measure distances between the pairs of observations, but will later enter into the calculations of that distance.

The Mahalanobis distance is similar to the Euclidean distance, except it weights the observations to account for the correlations between variables. By calculating the correlation matrix of the p variables as Z'Z/(n-1) (remembering Z is standardized), then the Mahalanobis distance for observation i from the origin is the square root of:

$$m_{ii} = (n-1)\underline{z}_i (Z'Z)^{-1} \underline{z}_i' \text{ and}$$

$$M = (n-1)Z(Z'Z)^{-1} Z' .$$

The n principal diagonal elements, the m_{ii}, are the squared Mahalanobis distances, again measured from the origin in the p-space. Note that if Z'Z/(n-1)=I, (i.e., the p variables are uncorrelated) the Mahalanobis distance is the Euclidean distance. Also note that if n<p, the solution is degenerate since Z'Z has rank less than p and so cannot be inverted. Principal components can be used to obtain approximate solutions by reducing the rank of Z'Z to something less than n (Chen (1974)).

Finally, the ranking procedure involves creating a new matrix K from the matrix Z. Each element in Z is transformed to its absolute value, so a new matrix Z* is created, where $z^*_{ij} = |z_{ij}|$. Then each column of Z* is transformed by ranking the observations in that column. The j^{th} column of K is the ranks of absolute values of the j^{th} column of Z. These ranks are summed across the rows, and the sum is divided by p, the number of variables. These results from K are the parametric distances.

Selection of Two Sites

Each of these measures can be used to determine a set of n distances from the centroid for the n observations. Initially it seems reasonable to take the observations with the minimum and second shortest distances from the centroid of the p-space as the combination of two observations which has the shortest total distance from the centroid. It follows that one would take the three observations with individual least distance from the centroid if choosing a combination of three observations to be most similar to the mean vector, and so on. However, it may be possible to find a combination of observations which has a smaller distance from the centroid than the combination which includes the two observations with

smallest distance individually. This would be done by combining, or averaging, two observations which are very dissimilar to each other. Positive values for the standardized variables would cancel out negative values between the two cases in the averaging process. The net result would be to obtain a combination which would be closer to the centroid.

This can be demonstrated by examining the squared Euclidean distance from the origin of the vector which is the average of vectors i and k (again from a mean vector of zeroes):

$$d_{\frac{i+k}{2}} = \sum_{j=1}^{p} (\tfrac{1}{2}(z_{ij} + z_{kj}) - 0)^2$$

$$= \tfrac{1}{4} \sum_j z_{ij}^2 + \tfrac{1}{4} \sum_j z_{kj}^2 + \tfrac{1}{2} \sum_j z_{ij} z_{kj}$$

$$= \tfrac{1}{4} d_{ii} + \tfrac{1}{4} d_{kk} + \tfrac{1}{2} d_{ik} . \qquad (1)$$

Contrast this to the distance between the same points i and k in the p-space:

$$d_{i:k} = \sum_j (z_{ij} - z_{kj})^2$$

$$= d_{ii} + d_{kk} - 2d_{ik} . \qquad (2)$$

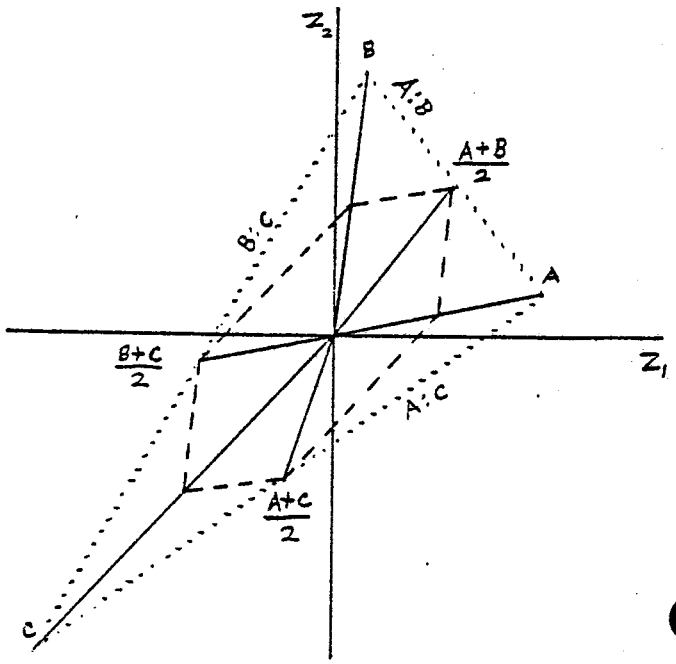
Combining equations (1) and (2) yields:

$$d_{\frac{i+k}{2}} = \tfrac{1}{2} d_{ii} + \tfrac{1}{2} d_{kk} - \tfrac{1}{4} d_{i:k} . \qquad (3)$$

That is, the squared Euclidean distance from the origin for the average of two observations is a simple linear function of their individual distances from the origin and the distance between the two cases. Since all the distances involved in (3) are nonnegative, it can be seen that the more dissimilar are two observations of constant length (the larger the distance between them), the smaller their combined distance from the centroid. Geometrically this can be shown as in Figure 1. (For details on the geometric presentation, see Davis (1967).) Three vectors, A,B, and C are shown in the Figure, with $d_{AA} < d_{BB} < d_{CC}$. The vector A is relatively close to B, however, and closer to C than C is to B, so $d_{A:B} < d_{A:C} < d_{B:C}$. Given these two circumstances, the vectors can be added as described above to obtain the three possible combinations of vectors A,B, and C to discover that $d_{\frac{B+C}{2}} < d_{\frac{A+C}{2}} < d_{\frac{A+B}{2}}$.

Note that this result is counter to what one would intuitively expect, as the length of the two longest vectors combined is shorter than the length derived from the two shortest vectors combined. In fact, combining the observations with the shortest and longest distances (A and C) also gives a shorter Euclidean distance from the origin than the combination involving the two shortest distances (A and B). Therefore, there is no guarantee that selection of the two observations in Z closest to the centroid will necessarily be the combination of observations which will yield an average vector having minimum distance from the centroid, except in the most fortuitous of circumstances.

Figure 1: Combination of Vectors in Two Dimensions



Note: The length of a vector $A = Z_{A1}^2 + Z_{A2}^2$ is simply the squared Euclidean distance of the point (Z_{A1}, Z_{A2}) from the origin.

To determine the distances from the origin of all sets of two observations, one can derive a matrix E:

$$E = \tfrac{1}{4} (\underline{d}\underline{u}') + \tfrac{1}{4} (\underline{u}\underline{d}') + \tfrac{1}{2} ZZ', \qquad (4)$$

where $\underline{d}$ is an n x 1 column vector consisting of the n elements from the principal diagonal of ZZ', and $\underline{u}$ is a n x 1 column vector with all entries equal to unity. The matrix E is symmetric, with entries along the principal diagonal equal to the Euclidean distances corresponding to the individual observations, and with off diagonal elements equal to the $d_{\frac{i+k}{2}}$ given in equation (1), the Euclidean distances of all the averaged pairs of observations. Note that where before there were n measures of distance, there are now $\frac{1}{2}(n)(n-1)$ distances to compare. Depending on the distances between the observations, any of the pairs may turn out to have an average vector with minimum distance from the origin.

General Selection of Multiple Sites

Now consider the situation where more than two sites are to be selected. By examining the combination of three observations, i, k, and m, one discovers:

$$d_{\frac{i+k+m}{3}} = \sum_{j=1}^{p} (1/3\ (z_{ij} + z_{kj} + z_{mj}))^2$$

$$= \frac{1}{9} d_{ii} + \frac{1}{9} d_{kk} + \frac{1}{9} d_{mm} + \frac{2}{9} d_{ik}$$

$$+ \frac{2}{9} d_{im} + \frac{2}{9} d_{km} .$$

And using an analogous argument to the two site case:

$$d_{\frac{i+k+m}{3}} = \frac{1}{3} d_{ii} + \frac{1}{3} d_{kk} + \frac{1}{3} d_{mm} - \frac{1}{9} d_{i:k}$$

$$- \frac{1}{9} d_{i:m} - \frac{1}{9} d_{k:m} .$$

Again the problem can be reduced to consideration of all distances and distances between observations. However, where before there were $\binom{n}{2}$ distances to examine, there are now $\binom{n}{3}$, an increase by a factor of $\frac{n-2}{3}$.

And instead of appearing in a vector or an array, these distances can be considered as entries in a cube, with the distances in cells along the principal diagonal still the individual Euclidean distances, (the principal diagonal is the set of cells containing d_{ikm}, where i = k = m). The off diagonal elements are the distances from the origin of the combinations of three observations to be examined. Off diagonal here refers to not only the four major diagonals in the cube, but also to the diagonals in the six faces of the cube. At this point matrix notation cannot be used, but a fast algorithm for examining cells in a multidimensional array can be developed for use in conjunction with the computer. Establishing and searching similar structures for distances for combinations of four or more observations is then handled in the same way.

Mahalanobis Distances

Establishing a similar argument for Mahalanobis distances is more difficult because of the weighting introduced by the correlations between the variables under consideration. There does not appear to exist any simple geometric way of presenting the results of the combination of two observations. Recall that the squared Mahalanobis distance from the centroid in the p-space for observation i is:

$$m_{ii} = (n-1)\ \underline{z}_i\ (Z'Z)^{-1}\ \underline{z}_i' .$$

In the special case of two variables with correlation r between them, m_{ii} can be easily calculated as:

$$m_{ii} = \left(\sum_{j=1}^{2} z_{ij}^2 - 2r\ z_{i1}\ z_{i2} \right) / (1-r^2)$$

$$= \frac{d_{ii} - 2r\ z_{i1}\ z_{i2}}{1 - r^2} .$$

Again, if r = 0, m_{ii} is equivalent to the Euclidean distance.

For two observations combined (i and k), the resultant squared distance $m_{\frac{i+k}{2}}$ of the average vector can be calculated as:

$$m_{\frac{i+k}{2}} = \frac{\frac{1}{4}(z_{i1}+z_{k1})^2 - \frac{1}{2}r(z_{i1}+z_{k1})(z_{i2}+z_{k2}) + \frac{1}{4}(z_{i2}+z_{k2})^2}{1-r^2}$$

$$= \frac{\frac{1}{4}d_{ii}+\frac{1}{4}d_{kk}+\frac{1}{2}d_{ik}-\frac{1}{2}r(z_{i1}+z_{k1}).(z_{i2}+z_{k2})}{1-r^2} .$$

Next, it is necessary to calculate the Mahalanobis distance between the two observations i and k, $m_{i:k}$, where:

$$m_{i:k} = \begin{pmatrix} z_{i1}-z_{k1} \\ z_{i2}-z_{k2} \end{pmatrix}' \begin{pmatrix} 1 & r \\ r & 1 \end{pmatrix}^{-1} \begin{pmatrix} z_{i1}-z_{k1} \\ z_{i2}-z_{k2} \end{pmatrix}$$

$$= \frac{d_{ii}+d_{kk} - 2d_{ik} - 2r(z_{i1}-z_{k1})(z_{i2}-z_{k2})}{1-r^2} .$$

Notice that the presence of the term d_{ik} implies that the Mahalanobis distance of the combination i and k from the origin will be a function of the distance between i and k. However, the rightmost terms involving $(z_{i1}+z_{k1})$ and $(z_{i2}+z_{k2})$ remain in the equation, so

$$m_{\frac{i+k}{2}} = \frac{\frac{1}{2}d_{ii} + \frac{1}{2}d_{kk} - \frac{1}{4}m_{i:k} - r(z_{i1} + z_{k1})(z_{i2}+z_{k2})}{1-r^2} \qquad (5)$$

In equation (5), because $(z_{ij}+z_{kj})$ could be either positive or negative, there is apparently no simple way to determine the net effect of combining cases i and k, as with Euclidean distance. With more than two variables under consideration, an analogue to equation (4) appears unlikely.

To obtain the desired Mahalanobis distances, calculated on all pairs of cases in the Z matrix, it is necessary to create a new matrix, T, which is of order $\binom{n}{q}$ x p, where each of the $\binom{n}{q}$ rows represents one of the $\binom{n}{q}$ combinations of q observations. For example, the first row of T would be the average of the first and second observations in Z. The second row of T would be the average of the first and third observations in Z, and so on.

This new matrix, T, can still be used to calculate the Mahalanobis distances using the Z'Z matrix, as the correlations between the p variables have not changed. The resultant $\binom{n}{q}$ x $\binom{n}{q}$ matrix:

$$M=(n-1)T\ (Z'Z)T' ,$$

contains on the principal diagonal the Mahalanobis distances from the origin for the $\binom{n}{q}$ combinations. Care must be taken after all the transformations have been made so the researcher can ascertain which rows in the matrix Z correspond to a particular row in the matrix T (or M).

Ranking

Analysis of the ranking procedure follows easily from the discussion of the Euclidean distances. Suppose two observations (i and k) are combined (averaged) as in Figure 1. The length of the resultant average vector is a function of the distances between the observations i and k and the lengths of i and k. Now the absolute value of each variable value is taken. The effect of this operation is to rotate all vectors back into the first quadrant, since all data values must now be positive. This affects only the direction, and not the length, of any of the combined vectors. Note this would not be the case if one were to take absolute values on all the elements in Z first, and then combine the vectors. The vectors must be combined before any distance calculations can be made, because the combining of the two vectors gives an average which represents the two items. By referring to the example in the section on Euclidean distances, the analysis of selection of multiple sites using ranking can be illustrated.

Referring to Figure 1, imagine vectors $\frac{A+B}{2}$, $\frac{A+C}{2}$, $\frac{B+C}{2}$ all rotated into quadrant I. Vector $\frac{B+C}{2}$ still has shortest length, and at least one of the variable values obtained by extending a perpendicular line from the end of the vector to each axis would necessarily be less than the corresponding variate values for vectors $\frac{A+C}{2}$ and $\frac{A+B}{2}$. So when ranks are taken, ranks averaged for the combination $\frac{B+C}{2}$ will yield minimum rank relative to $\frac{A+C}{2}$ and $\frac{A+B}{2}$ or a tie. There is no simple formulation which will yield the minimum combination. As in the case of the Mahalanobis distances, a new matrix of order $\binom{n}{q} \times p$ must be created, and then the ranking procedure outlined in an earlier section applied.

Conclusions

Selections of test sites and other related selection problems based on statistical distance measures are complex and may involve a good deal of preparation and calculation when more than one site is to be selected from a set. Choices based on individual characteristics or distances may not yield optimal groups of sites, as mutually dissimilar sites may in combination turn out to be offsetting, and the best choice to represent a region or the nation.

REFERENCES

1. Chen, H., Gnanadesikan, R., and Kettenring, J.R. (1974). "Statistical Methods for Grouping Corporations", Sankhyā, B36, 1-28.

2. Davis, Harry F. (1967). Introduction to Vector Analysis, Allyn and Bacon, Ind., Boston.

3. Gower, J.D. (1966). "Some Distance Properties of Latent Root and Vector Methods Used in Multivariate Analysis", Biometrika, 53, 325-338.

4. Kendall, M.G. (1975), Multivariate Analysis, Hafner Press, New York.

5. Mahalanobis, P.C. (1936). "On the Generalized Distance in Statistics", Proc. Nat Inst. Sci. India, 2, 49-55.

Smooth Curve Fitting with Shape preservation Using Osculatory Quadratic Splines

by

L. E. Deimel*
Dept. of Comp. Sci.
N. C. State U.
Raleigh, NC

D. F. McAllister**
Dept. of Comp. Sci.
N. C. State U.
Raleigh, NC

J. A. Roulier**
Dept. of Math.
N. C. State U.
Raleigh, NC

Abstract

Algorithms are presented for calculating an osculatory quadratic spline f. The spline f preserves the monotonicity and convexity of the data if the first derivatives are consistent with the monotonicity and convexity of the data. The knots of the spline are the data points and at most two additional knots between each adjacent pair of data points. The specification of the spline between two adjacent data points depends only on the points and the first derivatives at these points.

*Work supported in part by Rome Air Development Center, Contract F30602-75-C-0118
**Work supported in part by NSF Grant MCS76-0403

I. Introduction

There are several techniques for constructing smooth functions which interpolate data such as the method of ordinary polynomial interpolation [4], cubic spline interpolation [2], trigonometric interpolation [4], and convex spline interpolation [5,6,7]. There are also methods which use osculating cubic polynomials where the derivatives at each point are estimated using quadratic interpolation or a method proposed by Akima [1] which uses the slopes of the lines passing through adjacent points.

In all of the above methods, except for the convex spline methods, there is no guarantee that the monotonicity of the data is preserved by the interpolating function. The convex spline methods have the drawback that the data must first be divided into increasing, constant and decreasing segments and then further subdivided into convex and concave segments before the method can be applied. Although they do not require derivatives, they can be easily modified to use derivatives if they are available; however, the degree of the splines or the number of knots may be increased.

In this paper we propose a method for computing an osculating quadratic spline which interpolates data and first derivatives and preserves the monotonicity and convexity of the data if the first derivatives are consistent with the monotonicity and convexity. Moreover, the knots of these splines will be the data points and at most two additional knots between each adjacent pair of data points.

The method is useful in cases where first derivatives are available or can be estimated. It has the advantage over convex spline methods in that it uses only "local" information to compute the quadratic spline between two adjacent data points, namely the two points and the first derivatives associated with those points. For this reason it is easy to add and delete points and modify derivatives without having to recalculate all information required to specify the spline.

II. Notation and Background

It is well known that Bernstein polynomials inherit certain global properties of the functions they approximate, including monotonicity and convexity [5]. We will use these properties to develop an algorithm for computing a (smooth) osculating quadratic spline which preserves the monotonicity of given data. We will show that if the data are strictly decreasing and the given slopes are non-positive, then the quadratic spline will be strictly decreasing between

the data points.

Let $P = (P_1,P_2)$ and $Q = (Q_1,Q_2)$ be given points with $P_1 < Q_1$, and let g be a continuous function on the interval $I = [P_1,Q_1]$. The n-th degree Bernstein polynomial of g, $B_n(g)$, is defined as follows:

$$(2.1)\quad B_n(g)(x) = (Q_1-P_1)^{-n} \sum_{j=0}^{n} [g(P_1 + (j/n)(Q_1-P_1))$$
$$\binom{n}{j}(x-P_1)^j(Q_1-x)^{n-j}]$$

Let $W=(W_1,W_2)$ be an arbitrary point satisfying $P_1 < W_1 < Q_1$, let L_1 be the line through the points P and W with slope S_1, and let L_2 be the line through the points W and Q with slope S_2. Let g be the continuous piecewise linear function defined by the three points P, W, and Q (see Fig. 1):

$$g(x) = \begin{cases} L_1(x) \text{ for } x\in[P_1,W_1], \\ L_2(x) \text{ for } x\in[W_1,Q_1] \end{cases}$$

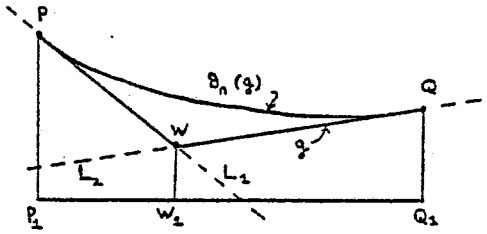


Figure 1

Define

$$(2.2)\quad n = \lceil (Q_1-P_1)/\min[(W_1-P_1),\ (Q_1-W_1)] \rceil \text{ and}$$
$$\phi = B_n(g).$$

Then ϕ satisfies the following on I:

(1) If g is increasing (decreasing) on I, then ϕ is increasing (decreasing) on I.

(2) If g is convex (concave) on I, then ϕ is convex (concave) on I.

(3) $\phi(P_1) = P_2$, $\phi(Q_1) = Q_2$

(4) $\phi'(P_1) = S_1$, $\phi'(Q_1) = S_2$.

We observe that if W_1 is the midpoint of I, then from (2.2), n=2 and $B_n(g)$ is a quadratic polynomial.

We will show in the following section that if the data are strictly decreasing ($P_2>Q_2$ with $S_1\le 0$ and $S_2\le 0$), then using the above properties of $B_n(g)$, it is always possible to specify a (smooth) quadratic spline on I which interpolates P, Q, S_1, and S_2 is strictly decreasing on I and requires at most two additional knots in I.

III. Algorithm Description

Let $P=(P_1,P_2)$ and $Q=(Q_1,Q_2)$ be points to be interpolated with $P_1<Q_1$ and $P_2>Q_2$. Let S_1 and S_2 be the slopes associated with P and Q, respectively, and L_1 and L_2 be the lines through P and Q with slopes S_1 and S_2, respectively. We assume in the sequel that $S_1\le 0$ and $S_2\le 0$. Let R be the set of points

$$R = \{(x,y): P_1<x<Q_1, \text{ and } Q_2<y<P_2\} - \{P,Q\}.$$

That is, R is the rectangle and its interior described by the points $A=(P_1,Q_2)$, $P=(P_1,P_2)$, $B=(Q_1,P_2)$, and $Q=(Q_1,Q_2)$ minus the points P and Q. Let M be the "midpoint" line through the points $F=((P_1+Q_1)/2,Q_2)$ and $G=((P_1+Q_1)/2,P_2)$. Let C and D be the points where L_1 and L_2 intersects the boundary of R, respectively. If L_1 and L_2 intersect, let $Z=(Z_1,Z_2)$ be a point of intersection (in R, if Z is not unique) (see Fig. 2).

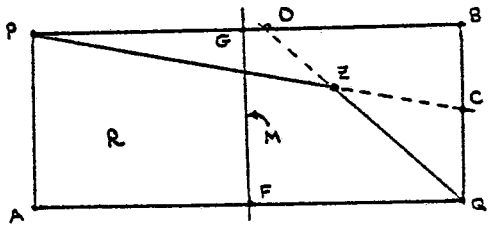


Figure 2

Case I: L_1 and L_2 intersect at a point Z in R (see Fig. 3).

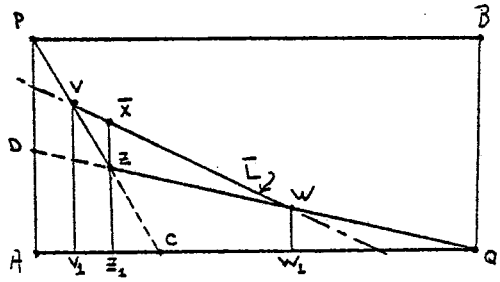


Figure 3

For this case the algorithm is as follows:

Let

(1) $\bar{X}_1 = Z_1$

(2) $V_1 = (P_1+\bar{X}_1)/2$,
$V_2 = L_1(v_1)$ ($V=(V_1,V_2)$ is the point on L_1 with abscissa V_1)

(3) $W_1 = (\bar{X}_1+Q_1)/2$,
$W_2 = L_2(W_1)$ ($W=(W_1,W_2)$ is the point on L_2 with abscissa W_1)

Let $\bar{L}$ be the line through the points V and W, and define

(4) $\bar{X}_2 = \bar{L}(\bar{X}_1)$

(5) q_1 to be the quadratic Bernstein polynomial

of the piecewise linear function defined by the three points P, V, and X (henceforth denoted by

$q_1 = B_2\{P,V,\overline{X}\}$ and

(6) $q_2 = B_2\{\overline{X},W,Q\}$, the corresponding quadratic Bernstein polynomial of the piecewise linear function determined by $\overline{X}$, W, and Q.

Now define f as follows:

$$f(x) = \begin{cases} q_1(x) \text{ for } x\varepsilon[P_1,\overline{X}_1] \\ q_2(x) \text{ for } x\varepsilon(\overline{X}_1,Q_1] \end{cases}$$

Then f is a decreasing, smooth osculatory quadratic spline with knot at $\overline{X}_1$, satisfying $f(P_1) = P_2$, $f(Q_1) = Q_2$, $f'(P_1) = S_1$ and $f'(Q_1) = S_2$.

It follows from the properties of Bernstein polynomials that f interpolates P, Q, and $\overline{X}$ and that f' interpolates S_1, S_2 and the slope of $\overline{L}$ at the points P, Q, and $\overline{X}$ respectively. Hence, it suffices to show that f is decreasing. This, however, is an immediate consequence of the fact that a Bernstein polynomial of a monotone function is monotone in the same sense, and that the above mentioned piecewise linear functions are decreasing. We note that in this case the spline f preserves the convexity of the piecewise linear function defined by the points P, Z, and Q.

Case II. L_1 and L_2 do not intersect.

There are three subcases:

Case IIa. Both L_1 and L_2 intersect the midpoint line M between the points F and G (see Fig. 4), or $S_1=S_2=0$.

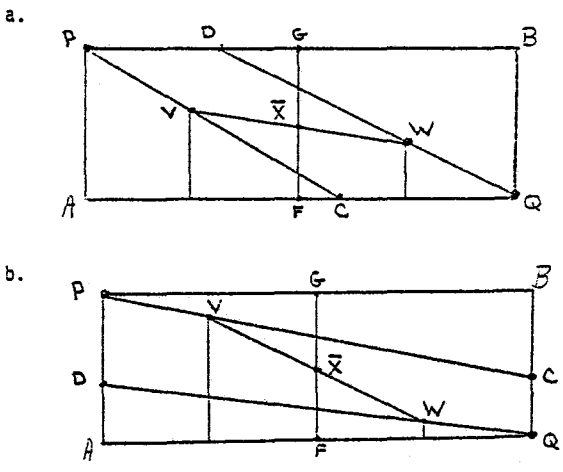


Figure 4

The algorithm is as follows:

Let

(1) $\overline{X}_1 = F_1$

Now proceed as in Case I.

The choice of $\overline{X}_1$ is not unique but the above choice insures a symmetry in the algorithm: if R were rotated interchanging P and Q, the algorithm applied and then the resulting graph rotated back, the function would be the same as if the algorithm had been applied without rotating R.

As in Case I, f is an osculatory quadratic spline. To show f is decreasing, it suffices to show that $V_2>W_2$. If L_1 lies above L_2 (see Fig. 4 (b)), then $V_2>W_2$ is obvious. If L_1 lies below L_2 in R (see Fig. 4 (a)), then by construction we have $V_2>(G_2+F_2)/2>W_2$ since both L_1 and L_2 cross M on the interior of R. Hence, L is decreasing and therefore f is also.

Note that in Cases IIb and IIc that C must lie on $\overline{AQ}$ and D must lie on $\overline{PB}$.

Case IIb. Only one of L_1 and L_2 intersects M on the interior of R.

Without loss of generality, assume that L_1 is the line which crosses M on the interior of R.

We construct two additional lines:

(a) Let H be the point on the line through A and Q so that C bisects the line segment $\overline{AH}$. Define $\hat{L}_1$ to be the line through P and H.

(b) Let J be the point on the line segment through P and B such that D bisects $\overline{JB}$. Let $\hat{L}_2$ be the line through Q and J (see Fig. 5).

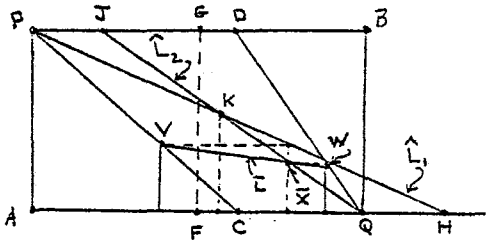


Figure 5

Define K to be the point of intersection of $\hat{L}_1$ and $\hat{L}_2$ (K lies in the interior of R).

Define

(1) $\overline{X}_1 = (K_1 + Q_1)/2$.

Now proceed as in Case I.

The knot $\overline{X}_1$ can be chosen arbitrarily in the interval (K_1,Q_1). The above choice insures symmetry in the algorithm as in Case IIa. (We note that it is also possible to choose $\overline{X}_1 = K_1$ but then the resulting spline will have zero slope at $\overline{X}$.)

To show that the line L is decreasing we note that by choice of $\overline{X}_1$ we must have

$V_2 = \hat{L}_1(\overline{X}_1)$ and

$W_2 = L_2(\overline{X}_1)$.

Since $\hat{L}_1$ lies above $\hat{L}_2$ to the right of K_1, it follows that $V_2 > W_2$ and hence f is decreasing.

Case IIc. Neither L_1 nor L_2 intersect M in the interior of R.

This is the only case in which we add two knots $\overline{X}_1$ and $\overline{\overline{X}}_1$ in the interval $[P_1,Q_1]$. In fact, except in the case that D = G and C = F, two such knots are required. That there exists no quadratic spline with a single knot for this case follows from Theorem 1 of [7]. Among other things, the theorem shows that the tangent lines to any quadratic polynomial at the end points of a closed interval [a,b] must intersect at x=(a+b)/2.

We also note that the lines $\hat{L}_1$ and $\hat{L}_2$ do not intersect in R and the triangles PCH and QDJ have no points in common (see Fig. 6).

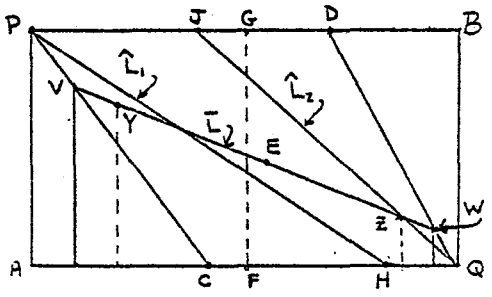


Figure 6

For this case the algorithm is as follows:

Let

(1) $\overline{X}_1 = (P_1 + C_1)/2$

(2) $V_1 = (P_1 + \overline{X}_1)/2$,
$V_2 = L_1(V_1)$

(3) $\overline{\overline{X}}_1 = (D_1 + Q_1)/2$

(4) $W_1 = (Q_1 + \overline{\overline{X}}_1)/2$,
$W_2 = L_2(W_1)$.

Define $\overline{L}$ to be the line through the points V and W.

Let

(5) $Y_1 = \overline{X}_1$,
$Y_2 = \overline{L}(Y_1)$

(6) $Z_1 = \overline{\overline{X}}_1$,
$Z_2 = \overline{L}(Z_1)$

and $E=((\overline{X}_1 + \overline{\overline{X}}_1)/2, \overline{L}((\overline{X}_1 + \overline{\overline{X}}_1)/2))$, the midpoint of the line segment $\overline{YZ}$.

Let $q_1 = B_2\{P,V,Y\}$,
$q_2 = B_2\{Y,E,Z\}$, and
$q_3 = B_2\{Z,W,Q\}$,

(Note that q_2 is the line $\overline{L}$). Define

$$f(x) = \begin{cases} q_1(x) \text{ for } x\in[P_1,\overline{X}_1], \\ q_2(x) \text{ for } x\in[\overline{X}_1,\overline{\overline{X}}_1], \\ q_3(x) \text{ for } x\in[\overline{\overline{X}}_1,Q_1]. \end{cases}$$

That f is decreasing follows from the fact that

$V_2 > (F_2+G_2)/2 > W_2$.

Hence, by construction and the fact that the second degree Bernstein polynomial of a line is the line, we have f is a decreasing quadratic spline which interpolates P and Q and f^1 interpolates S_1 and S_2.

In the special case that D = G and C = F, although it is possible to have only a single knot (anywhere between P_1 and Q_1), the resulting spline will have a zero slope at the knot, a condition which the authors wished to avoid. Hence this case was treated using two knots instead of one.

Finally, it remains to describe the algorithm for arbitrary data with arbitrary first derivatives. For increasing data ($P_2<Q_2$) with non-negative first derivatives we simply rotate the rectangle R about the line M and apply the algorithms already described. In all other cases, if L_1 and L_2 intersect at a point Z satisfying $P_1<Z_1<Q_1$, apply the algorithm of Case I; otherwise, apply the algorithm given in Case IIa.

IV. Numerical Results

The algorithm was applied to three examples considered by Akima [1] using his method for estimating derivatives at each point. The examples were as follows:

(1)	y	10	10	10	10	10	10	10.5	15	20	60	85
	x	0	1	2	3	4	5	6	7	8	9	10

Examples (2) and (3) used the same y values as example (1) but with the following x values:

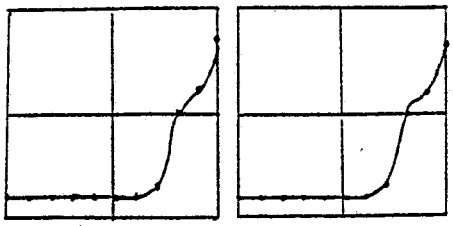
(2) 0,1,3,4,6,7,9,10,12,13,15 and

(3) 0, 2, 3, 5, 6, 8, 9, 11, 12, 14, 15.

The data is increasing beginning at y=10.5 in all cases. Since Akima's method produces non-negative slopes for increasing data, the quadratic spline will be strictly increasing beyond y=10.5.

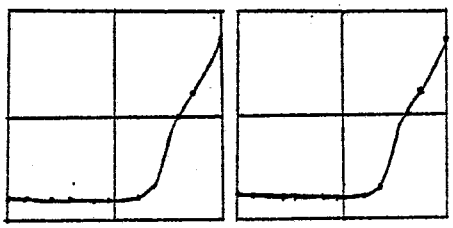
In examples (1) and (2) the graphs produced by the two methods were almost identical. However, in example (3), it can be shown that between the points (12,20) and (14,60) Akima's osculating cubic polynomial is decreasing at x=13 while the quadratic spline is strictly increasing through the interval. We should note that this was the only interval in all three examples where two knots were required for the quadratic spline. However, it is possible to redefine the slope at (12, 20) so that the resulting quadratic spline requires only one knot between each pair of data points. The graphs are given in Fig. 7.

(1)



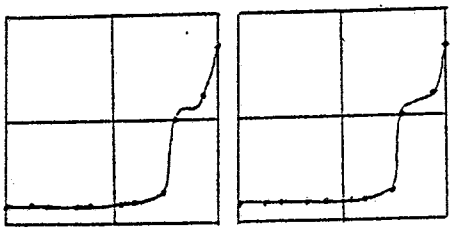
cubic quadratic

(2)



cubic quadratic

(3)



cubic quadratic

Figure 7

At the conference the authors will demonstrate the usefullness of the technique for general curve representation and compare the technique with Akima's for generation of random numbers from probability distribution functions as described in [3].

Acknowledgements: The authors would like to acknowledge the efforts of C. L. Doss II and Mark Evans in implementing the algorithms and helping with the figures. We would also like to thank Dr. Richard A. Tapia of Rice University for his useful discussions on the possible applications to random number generators.

References

1. H. Akima, "A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures", J. Assoc. Comp. Mach., Vol. 17, No. 4, Oct., 1970, pp. 589-602.

2. T. N. E. Greville, "Spline functions, interpolation, and numerical quadrature", in Mathematical Methods for Digital Computers, Vol. 2, A. Ralston and H. S. Wilf (eds.) Wiley, NY, 1967, Chapter 8.

3. V. O. Guerra, R. A. Tapia, and J. R. Thompson, "A Random Number Generator for Continuous Random Variables Based on an Interpolation Procedure of Akima", ICSA Technical Report, Rice University, 1972.

4. C. Lanczos, Applied Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1956.

5. D. F. McAllister, Eli Passow and J. A. Roulier, "Algorithms for Computing Shape Preserving Spline Interpolations to Data", Math. Comp., Vol. 31, 1977, pp. 717-725.

6. D. F. McAllister and J. A. Roulier, "Interpolation by Convex Quadratic Splines", Math. Comp. (Submitted)

7. Eli Passow and J. A. Roulier, "Monotone and Convex Spline Interpolation", SIAM J. Num. Anal., Vol. 14, 1977, pp. 904-909.

FOSOL: A LANGUAGE FOR STATISTICAL COMPUTING, MATRIX ALGEBRA AND TOP-TO-BOTTOM STRUCTURED PROGRAMMING

D. Anton Florian, Sangamon State University, Springfield, Illinois

ABSTRACT

FOSOL, a new programming language is described. It features interactive and batch modes of access, dynamic allocation of arrays, matrix algebra-type arithmetic statements, structured programming sequence control statements and an extensive run-time library of functions and procedures that plot, partition, concatenate, generate and analyze data structures.

1. INTRODUCTION

1.1 Objectives of FOSOL

FOSOL is a general purpose computing language that stresses statistical and matrix algebra facilities. The objective of FOSOL is to provide users of statistical and mathematical methods a natural and convenient way of expressing their problems. FOSOL was originally designed for instructional use as a tool in courses ranging from pre-calculus introductory statistics for non-statistics majors to multivariate analysis for statistics majors. FOSOL has been found to be useful in teaching linear algebra. In addition FOSOL has been used by statistical researchers to implement "state of the art" statistical analysis procedures.

FOSOL attempts to combine the simplicity of BASIC, the clarity of ALGOL 60 and PASCAL with the power and conciseness of APL. However, FOSOL hopes to go beyond these languages by providing an extensive run-time library of mathematical and statistical procedures. A large number of these procedures are, to our knowledge, not available elsewhere.

1.2 Implementation

FOSOL has been installed on two Control Data Corporation machines: Mid-Illinois Computer Cooperative's CYBER 72 under KRONOS and the University of Illinois' CYBER 175 under NOS. FOSOL is implemented as a preprocessor to FORTRAN. The preprocessor generates a FORTRAN driver program that is compiled and linked to the FOSOL run-time library. Nearly all of the processing in FOSOL is performed by the library functions and subroutines. Both the preprocessor and run-time library were designed and coded by this author in FORTRAN IV. Because portability for FOSOL was an important design criterion, FORTRAN was used. The machine dependent features of the preprocessor and the library that relate to the computer's word length, the input/output operations and the file system have been confined to a small set of primitive operations.

1.3 Documentation

One of the outstanding features of FOSOL is its extensive user oriented documentation: a bound manual by Florian [2] and an interactive on-line manual.

The on-line manual consists of a keyword driven, hierarchically structured text file residing on a word addressable random access file. Users may randomly page and read the text at their terminals or dispose portions of the text to their remote job entry terminal's high speed printer. The manual contains about 20,000 lines of text. Topics such as the initial log-on procedure for novices, permanent file commands, how to execute FOSOL in any of its three modes, language descriptions, hundreds of sample programs complete with output to illustrate the language and the extensive run-time library. The manual is written in a tutorial style with extensive discussions on methodology and computational procedures.

1.4 A Summary of the Features of FOSOL

To give an idea of the scope of FOSOL, we give a brief list of its features. Because of space limitations, only a few of these features are expanded on in this paper. Interested readers may consult reference [2].

FOSOL can be run in the interactive, batch and deferred batch modes. The language is the same in all three modes.

Extensive format-free, formatted and binary input/output operations on external files are available.

Complete syntax error checking with clear and meaningful error messages is provided. In the event of a run-time error, explicit reference to the user's statement is used to indicate where and why an error has occurred.

FOSOL is a structured programming language. A rich set of conditional, branching and iterative sequence control statements is provided. While FOSOL has no GOTO statement, two restricted unconditional transfer statements are given. FOSOL requires the user to write programs in a top-to-bottom structured programming style.

The present version of FOSOL has three numerical data types: integer, real and arrays. Arrays may be in rectangular or lower triangular form (for symmetric matrices).

Memory for arrays is dynamically allocated at run-time when they are assigned, generated or read from external files. FOSOL's memory manager performs garbage collection and compaction when needed. If necessary, the memory manager automatically increases the field length of the job (up to the user's validated limit). If this is not enough memory, FOSOL's virtual memory routines are called to manage the paging between the core memory work space and a word addressable random access file.

All arithmetic operators are polymorphic. Mixed mode arithmetic is permitted with all data types. All matrix algebra operators, including generalized inverses and tensor products are implemented.

Many of the functions such as the square root, logarithm and exponential are generic.

An extensive library of functions and procedures transform, plot, partition, concatenate, generate and analyze data matrices. The cumulative distribution function and inverse functions of all common discrete and continuous random variables are included. Many of the "state of the art" statistical procedures are not available elsewhere. The number and type of all arguments in the user's use of functions and procedures are checked. Whenever possible, FOSOL automatically performs data type conversions.

All permanent file commands and other KRONOS or NOS control language statements related to the job may be imbedded in the FOSOL program file. In the event of an error, the relevant portion of the user's dayfile is automatically listed.

User supplied FORTRAN and COMPASS subroutines and functions may be part of a FOSOL program file. The FOSOL processor has these routines compiled by the FTN FORTRAN compiler and linked to the FOSOL object program.

1.5 Future Plans

At the present time we are designing an expansion of the existing data structures and the required supporting library routines. The new data structures will include n-dimensional real arrays (and their lower triangular counterparts), n-dimensional boolean arrays, character string constants, variables and arrays. The booleans will permit the construction of generalizations of ALGOL 60 type conditional arithmetic and character string expressions. Also, we look forward to an opportunity to implement FOSOL on other machines.

2. LANGUAGE

FOSOL has two types of statments: simple and structured. Simple statements perform given tasks such as assignment, procedure calls and input/output operations. Structured statements such as conditional, branching and iterative statements control the order that the embedded simple statements are executed by the computer.

FOSOL statements are format free. Any number of statements may be placed on a line. A semicolon is used to denote the end of a statement. No special punctuation is needed to continue a statement to the next line.

2.1 Simple Statements

2.1.1 Assignment Statement

The assignment statement has form

```
variable := expression;
```

where "variable" is a name starting with an alphabetic character followed by any number of alphanumeric characters, ":=" is the assignment operator and "expression" is a constant, a variable, a function or an arithmetic expression involving constants, variables or functions.

The present version of FOSOL contains three numerical data types: integers, reals and arrays. The type of each newly defined variable is implicitly determined by its usage. For example, the statements

```
X1 := 123;
X2 := 12.3;
X3 := 1 2 3.4;
```

implicitly define X1, X2 and X3 of type integer, real and array respectively. After execution, the three variables contain the integer 123, the real number 12.3 and the 3 by 1 dimensional array

$$\begin{bmatrix} 1 \\ 2 \\ 3.4 \end{bmatrix}$$

respectively. The elements in an array or matrix may be either integer or real constants that are separated by blanks or commas. Matrix constants, other than column vectors, may be assigned by indicating the number of columns in parentheses after the assignment operator. For example, the statement

```
Y := (3) 1 2 3 4 5 6;
```

assigns to Y the matrix

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} .$$

FOSOL also allows assignments and operations on symmetric matrices in their lower triangular form to make more efficient use of the computer's resources.

2.1.2 Library Functions and Procedures

Constant and random matrices may be generated using the functions in the FOSOL library. For example, the statement

```
A := CON(C, N, M);
```

assigns to A an N by M matrix such that each element has value C (C, N and M may be scalar constants, variables or arithmetic expressions where the truncated values of N and M are used as dimensions). The statement

```
B:=RAN(100, 10, 50, 9);
```

assigns to B a 50 by 9 random matrix such that each element is a normal random number with a mean of 100 and a standard deviation of 10. A large number of deterministic and random matrix generators are available.

Output may be sent to the systems output file (an interactive user's terminal or a batch user's high speed printer) or any other designated file with the use of the write statments or with statistical analysis or plotting procedures that generate output. To illustrate the form of a FOSOL program and its output, we present a three statement program as it would be typed on an interactive terminal:

```
10  { PROGRAM GO ILLUSTRATE THE STRUCTURE OF A FOSOL PROGRAM
20    BY ANALYZING A DATA SET WITH THE PROCEDURE "CFRE".
30  }
40  X := 33 32 38 40 41 39 50 25 36 41 41 51 53 34 40 36 51 38
50  48.1 25 35.8 51 46 33 45 26.4 44.9 48 37 47.3;
60  PRINT "THE DATA SET X  IS",X;
70  CFRE(X);
```

The line numbers (digits on the far left of each line) are optional. Comments (character strings delimited by the square brackets) may appear anywhere. The PRINT statement, such as the one on line 70, may contain character strings, variable names and arithmetic expressions. The output from this sample program is given in Figure 1:

Figure 1

```
THE DATA SET X  IS

TRANSPOSED
  I = 30      1          2          3          4          5
    1      33.000     32.000     38.000     40.000     41.000
           39.000     50.000     25.000     36.000     41.000
           41.000     51.000     53.000     34.000     40.000
           36.000     51.000     39.000     48.100     25.000
           35.800     51.000     46.000     33.000     45.000
           26.400     44.900     48.000     37.000     47.300

VARIABLE NO.  1
                                FREQUENCY          RELATIVE FREQUENCY
CLASS      BOUNDARIES         COUNT  RELATIVE     HISTOGRAM
-------------------------------------.----.----.----.----.----.----.
  1   25.00     -  30.60        3   .100 .*****
  2   30.60     -  36.20        7   .233 .***********
  3   36.20     -  41.80        9   .300 .***************
  4   41.80     -  47.40        4   .133 .******
  5   47.40     -  53.00        7   .233 .***********
-------------------------------------.----.----.----.----.----.----.----.
                                      .0   .1   .2   .3   .4   .5   .6

SAMPLE SIZE    MEAN     STANDARD DEVIATION     VARIANCE       MEDIAN
     30       40.217          7.9830            62.141        38.379
```

Most FOSOL procedures have one required argument and several optional arguments that are used to direct the analysis. With CFRE, the column number of the data matrix to be analyzed, the number of classes for the frequency table and histogram and a window for the class limits may explicitly be stated.

The FOSOL library procedures range from simple descriptive statistical procedures, such as CFRE above, to general linear model analysis procedures such as GENOVA. The procedure GENOVA can alalyze all possible and multivariate experimental design models. There may be any number of factors (fixed effects or random effects from an infinite or finite population) at any number of levels in the model. The factors may have any possible nesting or hierarchical structure. There may be any number of replicates or missing cells, there may be any number of covariates and any number of dependent variables in the model. The procedure utilizes some unique operations available in FOSOL: generalized inverses, kronecker or tensor products and the dynamic storage allocation techniques that allow array structures such that each element in an array is itself also an array. These element arrays need not be of the same dimension.

To illustrate how conveniently a complex general linear hypotheses may be stated in a FOSOL program we give an illustration using the regression procedure REGG. The procedure REGG is a generalization of the analysis described in Searle [3]. REGG accepts design matrices that are not of full rank. It accepts any estimable restrictions on the model (the procedure checks if the restrictions are estimable). It accepts any testable linear hypotheses (the procedure checks for testibility).

In the example we assume the linear model

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \varepsilon_i$$

where y_i is an observation on the dependent variable, x_{i1}, x_{i2}, and xi_3 are observations on the independent variables, ε_i an error term and β_0, β_1, β_2 and β_3 are unknown parameters. In the sample run of REGG we impose the restriction $\beta_1 + \beta_2 = 1$ while simultaneously testing the hypothesis $\beta_3 = 0$. In matrix form the restriction and hypotheses may be stated in the form

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

The corresponding FOSOL restricted and hypothesized model matrix, say M, has form

$$M = \left[\begin{array}{cccc|c} 0 & 1 & 1 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 & 0 \end{array}\right].$$

In general, any number of restrictions of the form $R'\beta = C_1$ and hypotheses of the form $L'\beta = C_2$ may be input to REGG with an M matrix of the form

$$M = \begin{bmatrix} R' & C_1 \\ L' & C_2 \end{bmatrix}$$

where β is the vector of unknown parameters, R is a matrix of coefficients on the restrictions, L is a matrix of coefficients on the hypotheses and C_1 and C_2 are vectors of the appropriate dimensions. In Figure 2 we give both the FOSOL program and its output.

The procedure REGG assumes that the last (right most) column contains the observations on the dependent variable. The third parameter in REGG is used to denote the number of rows in the model matrix that are restrictions. Only the default output of REGG is presented here. REGG optionally prints the estimated covariance matrices of both the unconstrained and restricted/hypothesized models, residuals, residual plots and other statistics. Also, REGG optionally assigns to user supplied matrix names the least squares estimates vector, covariance matrix and vector of residuals. These output quantitites are available for subsequent calculations or input to other procedures in the program.

Data transformations are particularly simple to perform in FOSOL. For example, functions such as ABS (for absolute value), SIN (for trigonometric sine) and SQRT (for square root) are generic. They accept integer, real and matrix arguments and return integers, reals and matrices. These functions are very powerful for matrix arguments. For instance, if A is a matrix then the statement

```
B := SQRT(A);
```

generates a new matrix B of the same dimension as A where the elements of B are the square roots of the corresponding elements of A. Each generic function has an optional second scalar argument that can be used to select a given column or row of the argument matrix to operate on. For example, the statement

```
B := SQRT(A,K);
```

limits the square root operation to the K^{th} column of A if K is positive and to the $-K^{th}$ row of A if K is negative. If, in the above statement B is replaced by A, then the statement essentially performs a row or column operation on matrix A; a new matrix is not generated.

In addition to these generic functions, the FOSOL library contains over 200 user-callable functions and procedures that plot, transform, concatenate, lag and smooth matrices. Included are generalization of the Beaton [1] operators, such as our generalized sweep, Cholesky and eigenvalue decompositions. Also, all the common discrete probability functions and discrete and continuous cumulative distribution functions are in the library.

Figure 2

```
10 [ PROGRAM TO ILLUSTRATE THE PROCEDURE "REGG". ]
20 M:=(5) 0 1 1 0 1  0 0 0 1 0;  [ DEFINE THE MODEL MATRIX ]
30 D:=(4) 1 2 3 3.5  3 3 3 4.5  3 4 5 5  5 4 3 5
40     8 6 9 6.6  4 3 7 4  3 4 2 6;
50 PRINT "MODEL MATRIX",M,"DATA MATRIX",D;
60 REGG(D,M,1);
```

OUTPUT:

MODEL MATRIX

2 * 5	1	2	3	4	5
1	0.	1.0000	1.0000	0.	1.0000
2	0.	0.	0.	1.0000	0.

DATA MATRIX

7 * 4	1	2	3	4
1	1.0000	2.0000	3.0000	3.5000
2	3.0000	3.0000	3.0000	4.5000
3	3.0000	4.0000	5.0000	5.0000
4	5.0000	4.0000	3.0000	5.0000
5	8.0000	6.0000	9.0000	6.6000
6	4.0000	3.0000	7.0000	4.0000
7	3.0000	4.0000	2.0000	6.0000

** FOSOL GENERAL LINEAR HYPOTHESES - REGG **

UNCONSTRAINED MODEL

ANALYSIS OF VARIANCE

SOURCE OF VARIATION	DEGREES OF FREEDOM	SUM OF SQUARES	MEAN SQUARE	F-RATIO	P-VALUE
MEAN	1	171.02	171.02	1471.2	.00003
REGRESSION	3	6.6884	2.2295	19.178	.01830
RESIDUAL	3	.34875	.11625		
TOTAL	7	178.06			

R - SQUARE: .95044
STANDARD ERROR OF THE REGRESSION: .34095

LEAST SQUARES ESTIMATES

VARIABLE NO.	ESTIMATE	STANDARD ERROR	T-STATISTIC	P-VALUE
0	1.7362	.55064	3.1531	.04978
1	-.12265	.17153	-.71504	.52870
2	1.1231	.25249	4.4481	.01950
3	-.10757	.80827E-01	-1.3309	.27563

HYPOTHESIZED AND/OR RESTRICTED MODEL

ANALYSIS OF VARIANCE

SOURCE OF VARIATION	DEGREES OF FREEDOM	SUM OF SQUARES	MEAN SQUARE	F-RATIO	P-VALUE
RESTRICTIONS	1	.10833E-05			
FULL MODEL	3	177.71	59.237	1528.7	.00004
HYPOTHESES	1	.22367	.22367	1.7240	.25962
REDUCED MODEL	2	177.49	88.744		
RESIDUAL	3	.34875	.11625		
TOTAL	7	178.06			

ANALYSIS OF VARIANCE

SOURCE OF VARIATION	DEGREES OF FREEDOM	SUM OF SQUARES	MEAN SQUARE	F-RATIO	P-VALUE
REDUCED MODEL	2	177.49	88.744	775.16	.00003
REDUCED ERROR	5	.57242	.11448		
TOTAL	7	178.06			

STANDARD ERROR OF THE REGRESSION: .33835

LEAST SQUARES ESTIMATES

VARIABLE NO.	ESTIMATE	STANDARD ERROR	T-STATISTIC	P-VALUE
0	1.2694	.12891	9.8466	.00060
1	-.28548	.11369	-2.5111	.05315
2	1.2955	.11369	11.307	.00026
3	0.	0.	0.	1.00000

2.1.3 Arithmetic Operators

One of the most powerful features of FOSOL are the polymorphic arithmetic operators. Mixed mode arithmetic is permitted with all data types. The arithmetic operators used in constructing all arithmetic expressions are:

- `+` for addition
- `-` for subtraction or negation
- `*` for multiplication
- `/` for division, reciprocal or inversion
- `'` for transpose multiplication or transposition
- `**` for exponentiation
- `!` for factorial operation.

The operators -, / and ' may be used as both unitary and binary operators. The expression -A yields the negative of A for all data types. If A is a scalear, /A yields a reciprocal of A. If A is a square matrix of full rank, /A yields the inverse of A; otherwise /A yields a generalized inverse of A. The expression A' yields the transpose of A. Operators may be combined. For example, if A is a matrix, the statement

```
B := -/A';
```

assigns to B the negative of the generalized inverse of the transpose of A. The operator ! may only be used as a post-fix unitary operator. The expression A! yields A factorial.

For matrix operands, the binary operators permit full matrix algebra type operations. For example, with the inversion operator / and the matrix product operator * solving a system of linear equations becomes a trivial problem. The system of equations (written in matrix algebra form)

$$AX = C$$

where A is the coefficient matrix, X the column vector of unknowns and C a column vector of constants may be solved with the statement

```
X:=/A*C;
```

The usual matrix algebra solution $X=A^{-1}C$ is obtained if A is square and of full rank. If A is not square or A is not of full rank no unique solution to the system exists. The FOSOL solution in this instance is one of the infinite number of possible solutions to the system.

There is virtually no limit to the complexity of the expressions that are admissible. For example, in general linear hypotheses theory for the model

$$Y = X\beta + \varepsilon,$$

as described above in connection with procedure REGG, the least squares estimate, $\hat{\beta}$, of the unknown vector β, subject to the condition $L'\hat{\beta}=C$, is given by the matrix algebra expression

$$\hat{\beta} = GX'Y - GL(L'GL)^{-}(L'GX'Y-C)$$

where $G = (X'X)^{-}$, which is a generalized inverse of X'X. This solution may be coded directly into two FOSOL statements:

```
G := /(X'X);

B := G*X'Y-G*L/(L'G*L)*(L'G*X'Y-C);
```

One of the outstanding features of FOSOL is its superlative compile-time and run-time diagnostics and error recovery systems. The syntax analyzer detects any misuse of operators, operands, keywords, special symbols, structured statements, the number and types of arguments in library functions and procedures. Virtually all run-time error conditions are trapped when they occur and informative messages are printed. The following program and its output illustrates the run-time error message system:

Figure 3

```
10 ! PROGRAM TO ILLUSTRATE THE RUN-TIME ERROR MESSAGE PRINTED
20   WHEN THE DIMENSIONALITY REQUIREMENT ON THE TWO MATRIX
30   OPERANDS IS NOT MET. !
40 A:=(2) 1 2 3 4;
50 B:=(3) 11 12 13 14 15 16 17 18 19;
60 PRINT "MATRIX A",A,"MATRIX B",B;
70 PRINT "TWICE THE PRODUCT OF A AND B",2*A*B;

OUTPUT:

MATRIX A

 2 * 2     1           2

   1      1.0000      2.0000
   2      3.0000      4.0000

MATRIX B

 3 * 3     1           2           3

   1      11.000      12.000      13.000
   2      14.000      15.000      16.000
   3      17.000      18.000      19.000

TWICE THE PRODUCT OF A AND B


 F A T A L   E R R O R   ON LINE

70 PRINT "TWICE THE PRODUCT OF A AND B",2*A*B;
                                        #


   THE OPERATION IS NOT DEFINED BECAUSE OF THE INCORRECT
   DIMENSIONS OF THE ARRAYS.   THEIR FIRST 5 ROWS ARE

    2 * 2     1           2

      1      2.0000      4.0000
      2      6.0000      8.0000


    3 * 3     1           2           3

      1      11.000      12.000      13.000
      2      14.000      15.000      16.000
      3      17.000      18.000      19.000
```

Notice that the operator "*" was flagged by the symbol "#" and the present value of both operands was printed. We feel that clear and unambiguous error messages are of primary importance in any computing environment.

2.2 Structured Statements

2.2.1 Introduction

A sequence of simple statments are always executed in the order that they appear in the program. For a large number of problems, considering the power of the FOSOL simple statements and the functions and procedures in the FOSOL library, a sequence of simple statements is sufficient. Structured statements are used whenever a straight sequence of simple statements cannot be used to solve a problem to the desired generality. A general solution to some problems often necessitates an examination the data which may require that certain simple statements be executed conditionally. It may also be desirable to execute a group of simple statements conditionally until one or more conditons are met. The structured statements in FOSOL add considerable power and flexibility to the language.

Structured statements have a common form: each statement type has a beginning, usually a keyword that denotes the type of structured statement, and an end, called the head and the tail of the structured statement respectively. The statements between the head and the tail of the structured statement are the body. The body may consist of a sequence of simple statements or other structured statements with their own heads, bodies and tails. Any structured statement that is nested inside another structured statement is treated as though it were a simple statement by the structured statement that it is nested in. The nesting may go any depth.

There are six structured statements: the DO, the IF, the CASE, the REPEAT, the WHILE and the BEGIN statements. In addition, there are two transfer statements, the NEXT and the EXIT, that transfer execution control to the head of a structured statement and to the statement following the tail of a structured statement respectively.

Of the six structured statements, four are repetitive statements: DO, REPEAT, WHILE and BEGIN. The DO statement is appropriate when the number of repetitions on the body is known beforehand. The REPEAT and WHILE are suitable for situations where the number of repetitions is not known beforehand. The BEGIN statement, when used in conjunction with the two transfer statements NEXT and EXIT, is a generalization of the other three repetitive statements and may be used when more generality is desired.

The IF statement is used to execute one or more simple or other structured statements conditionally. There are three variations of the IF statement - each having one, two or more than three sub-bodies that are executed depending on certain conditions. The CASE statement is an n-way branching statement that permits the execution of one of n simple or structured statements in the body of the CASE statement.

2.2.2 The DO Statement

The DO statement has form:

DO v := e_1 TO e_2 BY e_3;

body

END v;

where v is the iterative variable, e_1 is the initial value, e_2 is the terminal value and e_3 the increment. The part "BY e_3"optional; if omitted an increment of 1 is assumed. The type of the incremental variable v is determined by the type of e_1. The e_1, e_2 or e_3 may be constants, variables or arithmetic expressions. The increment may be either positive or negative.

The program in Figure 4 prints a short table of the cumulative standard normal distribution function and illustrates the DO statement:

Figure 4

```
10 DO Z := -4. TO 4. BY .5;
20    PRINT Z,CND(Z);
30 END Z;

OUTPUT:

 -4.0000        .31671E-04
 -3.5000        .23263E-03
 -3.0000        .13499E-02
 -2.5000        .62097E-02
 -2.0000        .22750E-01
 -1.5000        .66807E-01
 -1.0000        .15866
 -.50000        .30854
  0.            .50000
  .50000        .69146
  1.0000        .84134
  1.5000        .93319
  2.0000        .97725
  2.5000        .99379
  3.0000        .99865
  3.5000        .99977
  4.0000        .99997
```

2.2.3 The IF Statement

For the IF, WHILE and REPEAT statements, logical expressions are used to control the order of execution. The FOSOL relational operators used in consulting logical expressions are listed in Figure 5.

Figure 5

Relational Operators

Symbol	Meaning
=	equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
<>	not equal to

In addition, the three logical operators OR, AND and NOT can be used to construct logical expressions.

The short form of the IF statement is

```
IF ℓ THEN
   b
FI;
```

where ℓ is the logical expression and b is a body of simple or other structured statements. An example is

```
IF X<0 THEN
   PRINT "X HAS VALUE", X;
   X := -X;
FI;
```

The statements in the body are executed if the logical expression in the head is true, otherwise, they are skipped over. The most general IF statement has form:

```
IF ℓ1 THEN
   b1
ORIF ℓ2 THEN
   b2
ORIF ℓ3 THEN
   b3
.
.
.
ORIF ℓn THEN
   bn
ELSE
   bn+1
FI;
```

The ORIF heads and their bodies as well as the ELSE head and its body are optional. When they are present, each logical expression is evaluated in turn until a true logical expression, say ℓ_i is found. In this case only the body of statements b_i is executed; the statements in all other bodies are skipped over. If none of the logical expressions are found to be true, the statements in the ELSE body are executed.

The following is a part of a program that illustrates the use of the DO and IF statements to sort a vector X in ascending order using the bubble sort algorithem, and then computing the median (this is given for illustrative purposes only since the median library function could be used for this task):

```
N:=NROW(X);   [NROW returns the number of row in X]
DO I:=1 TO N-1;
    DO J:=I TO N;
        IF X(I) > X(J) THEN   [interchange]
            T:=X(I);
            X(I):=X(J);
            X(J):=T;
        FI;
    END J;
END I;



IF N=1 THEN
        PRINT "SAMPLE IS OF SIZE ONE";
        MED:=X(1);
    ORIF (N/2)*2=N THEN [N is even]
        MED:= (X(N/2)+X(N/2+1))/2;
    ELSE
        MED:=X(N/2+1);
FI;
PRINT "MEDIAN=", MED;
```

2.2.4 The WHILE and REPEAT Statements

The WHILE and REPEAT statements have a similar structure. Their forms are:

```
WHILE ℓ DO label;          REPEAT label;
    body                       body
END label;                 UNTIL ℓ;
```

where the ℓ are logical expressions and "label" are optional labels. In the WHILE statement the statements in the body are executed repeatedly until the logical expression in the head is false. In the REPEAT statement, the statements in the body are executed repeatedly until the logical expression in the tail is true. For example, in the following sequence of statements

```
REPEAT;
     X:=RAN(0,1,10,10);
UNTIL DET(X)>0.;
```

a 10 by 10 matrix X, where each element is a standard normal random number, is generated repeatedly until one with a positive determinant is found.

2.2.5 The NEXT and EXIT Statements

FOSOL has two unconditional transfer statements: NEXT and EXIT. Either statement may only be used as the last statement in an IF, ORIF or ELSE conditional block. The form of the statements are

```
NEXT label;  and  EXIT label;
```

The statements must also be nested in a DO, WHILE, REPEAT or BEGIN structured statement that has the same label used in the NEXT or EXIT statements. When the NEXT statement is executed, control is transferred to the head of the referenced structured statement and the body of that structured statement is executed again. When the EXIT statement is executed, control is transferred to the statement following the tail of the referenced structured statement. Both statements may be used to break out of nested structured statements. For example, in the following prime number generator program, the NEXT N statement breaks out of the inner DO I loop after N is found to have I as a factor:

```
DO N:=2 TO 20;
    DO I:=2 TO SQRT(N);
        IF(N/I)*I=N THEN
            NEXT N;
        FI;
    END I;
    PRINT N, "IS PRIME";
END N;
```

In the following example the EXIT I statement breaks out of the outer DO I loop after the first negative elements in the matrix A is found:

```
M := NCOL(A);
.DO I := 1 TO NROW(A);
     DO J:= 1 TO M;
          IF A(I,J)<0. THEN
               PRINT "NEGATIVE", I,J,A(I,J);
               EXIT I;
          FI;
     END J;
END I;
```

In the above example, NCOL(A) returns the number of columns in matrix A.

2.5.6 The BEGIN Statement

The BEGIN statement has form

```
BEGIN label

    body

END label;
```

With the use of the IF, NEXT and Exit statements, the BEGIN statement, with appropriate initialization, serves to generalize the DO, WHILE and REPEAT statements. The BEGIN statement also serves to collect a series of statements together so that the CASE statement considers these statements as a single case.

2.5.7 The CASE Statement

The CASE statement is a conditional n-way branch that does not need labels. The CASE statement has form:

```
CASE e OF;
     statement1;
     statement2;
       .
       .
       .
     statementn;
END;
```

where e is a scalar variable or arithmetic expression called the selector. The action of the CASE statement is as follows: If the selector is not an integer variable then it is evaluated and truncated to an integer value (if the selector expression is real). Suppose that the value of the selector is k. If the value of k is between 1 and n, then the k^{th} statement below the case head is executed. All other statements are skipped and execution continues with the statement following the END statement.

If k has value less than 1 or greater than n, then all n statements are skipped. The case statement has an optional ELSE structure (as in IF statements) that may be placed before the END statement. The statements in the ELSE structure are executed only if the above k value is less than 1 or greater than n.

The statements in the body of the case statement may be simple statements or structured statements. Each structured statement (complete with head and tail) is treated as a single statement as far as the case statement is concerned.

2.5.8 The SUM Expression

The operation of summation is basic to general formulation of many relationships in mathematics. While summations may be computed using the structured statements described thus far, the structured summation expressions are far more convenient. The FOSOL equivalent for the condition summation

$$\sum_{i=n}^{m} \text{condition}$$

is SUM I := N TO M FOR 2:

The "FOR 2", containing a logical expression 2, is optional. It may be used to restrict the inclusion of certain terms in the sum.

For example, the FOSOL equivalents for computing the mean, m and standard deviation s of the numbers:

$$x_1, x_2, \ldots, x_n$$

with the formulas

$$m=\frac{1}{n}\sum_{i=1}^{n} x_i \qquad \text{and} \qquad s=\sqrt{\frac{\sum_{i=1}^{n}(x_i-m)^2}{n-1}}$$

are

```
N := NROW(X);
M := (SUM I:=1 TO N: X(I))/N;
S := SQRT(SUM I:=1 TO N: (X(I)-M**2)/(N-1));
```

Sums are treated as arithmetic expressions and may be nested any depth. Thus, for example, if A is on n by m matrix, the FOSOL equivalent of the restricted sum of squares of the elements of A, as given by the double sum

$$s = \sum_{\substack{i=1 \\ i\neq j}}^{n} \sum_{j=1}^{m} a_{ij}^2 ,$$

is

```
N := NROW(A);
M := NCOL(A);
S := SUM I:=1 TO N : SUM J:=1 TO M
     FOR I <>J : B(I,J)**2;
```

The start and end for the summation index may be any scalar constant, variable or arithmetic expression.

REFERENCES

1. Beaton, A.E. "The Use of Special Matrix Operators in Statistical Calculus." Ed.D. Thesis, Harvard University. Reprinted as Educational Testing Service Research Bulletin, 64-51, Princeton, N.J.

2. Florian, D.A. FOSOL User Manual, Springfield, IL: Sangamon State University bookstore. 1977.

3. Searle, S.R. Linear Models, New York, John Wiley & Sons, 1971.

VARIANCE-COMPONENT ESTIMATION FOR THE UNBALANCED TWO-WAY RANDOM CLASSIFICATION

BY

F. Giesbrecht and Lynn Dix

North Carolina State University
Raleigh, North Carolina 27607

ABSTRACT

This paper presents a solution to the problem of computing minimum norm quadratic unbiased estimates (MINQUE) of variance components for the unbalanced two-way classification model with all factors random. An important intermediate result is a compact method for obtaining the inverse of the variance-covariance matrix of the vector of observations.

1. INTRODUCTION

This report presents a solution to the problem of computing minimum norm quadratic unbiased estimates (MINQUE) for the unbalanced two-way random classification model. The method originally described by Rao (1971a, 1971b, 1972) required the inversion of an n x n matrix for a data set with n observations, with $n = \Sigma n_{ij}$ and n_{ij} the number of observations in the cell corresponding to row i, column j of an r x c table. An alternative method, described by Hemmerle and Hartley (1973) and Liu and Senturia (1977) requires the inversion of an $(r+c+rc^*)x(r+c+rc^*)$ matrix when an interaction component is present in the model ($rc^* \leq rc$ is the number of non-empty cells) and the inversion of an (r+c)x(r+c) matrix when there is no interaction. The method developed here allows the interaction component in the model and yet requires only the solution of a system of r linear equations and a system of c linear equations. The resulting saving in computer space becomes appreciable when one considers data sets approaching the sizes encountered by some researchers (Lee, 1976).

A computer program (in Fortran) implementing the techniques derived in this paper has been developed. The emphasis has been on minimizing the amount of computer memory required in order to maximize the size of the data sets that can be processed.

2. MODEL AND NOTATION

The model assumed is

$$y_{ijk} = \mu + r_i + c_j + rc_{ij} + e_{ijk} \qquad (2.1)$$

for $k=1, \ldots, n_{ij}$, $i=1,\ldots,r$ and $j=1,\ldots,c$, where μ is a fixed constant, $\{r_i\}$, $\{c_j\}$, $\{rc_{ij}\}$ and $\{e_{ijk}\}$ are independent sets of independent identically distributed random variables with mean zero and variances σ_r^2, σ_c^2, σ_{rc}^2 and σ_e^2 respectively. More compactly, $\underset{\sim}{Y}$ denotes the $n = \Sigma n_{ij}$ observations in dictionary order. If $v^*_{ijki'j'k'}$ denotes a typical element in the variance-covariance matrix of $\underset{\sim}{Y}$ then it follows that

$$v^*_{ijki'j'k'} = \sigma_r^2\delta_{ii'} + \sigma_c^2\delta_{jj'} + \sigma_{rc}^2\delta_{ii'}\delta_{jj'} + \sigma_e^2\delta_{ii'}\delta_{jj'}\delta_{kk'} \qquad (2.2)$$

where

$$\delta_{hh'} \begin{cases} = 1 & \text{if } h = h' \\ = 0 & \text{if } h \neq h' \end{cases}.$$

In matrix notation one can write the variance-covariance matrix as

$$D(Y) = \sigma_r^2 V_r + \sigma_c^2 V_c + \sigma_{rc}^2 V_{rc} + \sigma_e^2 V_e \quad (2.3)$$

where the definitions of V_r, V_c, V_{rc}, and V_e follow from (2.2). The minimum norm quadratic unbiased (Rao 1971a, 1971b, 1972) or restricted maximum likelihood estimates if one assumes normality, (Harville 1977) are obtained by computing the four quadratic forms, $Y'Q_V V_i Q_V Y$ for i = r,c,rc and e, equating them to their expected values and solving for the four variance components, where

$$Q_V = V^{-1} - V^{-1}X_0(X_0'V^{-1}X_0)^{-1}X_0'V^{-1} ,$$

$$V = \alpha_r V_r + \alpha_c V_c + \alpha_{rc} V_{rc} + \alpha_e V_e ,$$

X_0 is an n x 1 matrix of ones and α_r, α_c, α_{rc} and α_e represent the prior guesses for the four variance components. If an iterative scheme is contemplated then in the m'th iteration the $\{\alpha_i\}$ are the estimates produced in iteration m-1.

It is immediately clear that computing the n x n matrix V^{-1} is the major hurdle in this method. Standard numerical inversion techniques rapidly become unreasonable as n becomes large. The object of this paper is to obtain the elements of V^{-1} by using a method that only requires solutions for a system of r linear equations and a system of c linear equations.

3. INVERSE MATRIX

The typical element of V^{-1} can be written as

$$v^{-1}_{ijki'j'k'} = w_{iji'j'} + wc_{iji'}\delta_{ii'} + wr_{ii'j}\delta_{jj'} + wrc_{ij}\delta_{ii'}\delta_{jj'} + \alpha_e^{-1}\delta_{ii'}\delta_{jj'}\delta_{kk'} .$$

The $\{w_{iji'j'}\}$, $\{wc_{iji'}\}$, $\{wr_{ii'j}\}$ and $\{wrc_{ij}\}$ are functions of the $\{\alpha_i\}$ and $\{n_{ij}\}$ and must satisfy the following four sets of equations:

Set 1.

$$\alpha_{rc}(1/\alpha_e) + (\alpha_{rc}n_{ij} + \alpha_e)wrc_{ij} = 0 \quad \forall\ i,j. \quad (3.1)$$

Set 2.

$$\alpha_c(1/\alpha_e) + \alpha_c n_{ij} wrc_{ij} + \alpha_c \Sigma_{i*} n_{i*j} wr_{i*ij} + (\alpha_{rc}n_{ij} + \alpha_e)wr_{ii'ij} = 0 \quad \forall\ i,i'j. \quad (3.2)$$

Set 3.

$$\alpha_r(1/\alpha_e) + \alpha_r n_{ij} wrc_{ij} + \alpha_r \Sigma_{j*} n_{ij*} wc_{ij*j} + (\alpha_{rc}n_{ij} + \alpha_e)wc_{iji'j'} = 0 \quad \forall\ i,j,j'. \quad (3.3)$$

Set 4.

$$\alpha_r n_{ij} wr_{ii'j} + \alpha_c n_{ij} wc_{iji'} + \alpha_r \Sigma_{j*} n_{ij} w_{ij*ii'j*} + \alpha_c \Sigma_{i*} n_{ij} w_{ij*i'j} + (\alpha_{rc}n_{ij} + \alpha_e)w_{iji'j'} = 0 \quad \forall\ i,j,i',j'. \quad (3.4)$$

From the nature of V and V^{-1} one can see that $wr_{ii'j} = wr_{i'ij}$ and $wc_{iji'} = wc_{ij'i}$ $\forall i,i',j,j'$.

Set 1 immediately yields

$$wrc_{ij} = -\alpha_{rc} / \left[\alpha_e(n_{ij}\alpha_{rc} + \alpha_e)\right] \quad \forall\ i,j.$$

Set 2 consists of rc subsets of r equations, i.e. a set of r equations for every pair (i,j). Solving these yields

$$wr_{i'ij} = -\alpha_c / \left[(n_{ij}\alpha_{rc} + \alpha_e)(n_{i'j}\alpha_{rc} + \alpha_e)\, q_j\right]$$

where

$$q_j = 1 + \Sigma_{i*}\left\{\frac{n_{i*j}\,\alpha_c}{(n_{i*j}\,\alpha_{rc} + \alpha_e)}\right\} .$$

Similarly set 3 yields

$$wc_{iji'} = -\alpha_r / \left[(n_{ij}\alpha_{rc} + \alpha_e)(n_{ij'}\alpha_{rc} + \alpha_e)\, p_i\right]$$

where $p_i = 1 + \Sigma_{j*}\left\{\frac{n_{ij*}\,\alpha_r}{(n_{ij*}\alpha_{rc} + \alpha_e)}\right\}$.

Set 4 consists of rc subsets of rc equations. For specific i' and j' the equations can be identified as

$$f_{ij}: \alpha_r \sum_{j*} n_{ij*} w_{ij*i'j'} + \alpha_c \sum_{i*} n_{i*j} w_{i*ji'j'}$$

$$+ (\alpha_{rc} n_{ij} + \alpha_e) w_{iji'j'}$$

$$= - \alpha_r n_{ij'} w r_{ii'j'} - \alpha_c n_{i'j} w c_{i'jj'} \quad .$$

In order to solve these equations a number of linear operations are performed. For all $i < r$ and $j < c$ replace f_{ij} by $f_{ij} - f_{rj} - f_{ic} + f_{rc}$, f_{ic} by

$$f_{ic} - f_{rc} - \alpha_r \sum_{j*}^{c-1} \left\{ \frac{n_{ij*} f_{ij*}}{n_{ij*} \alpha_{rc} + \alpha_e} \right\}$$

and f_{rj} by

$$f_{rj} - f_{rc} - \alpha_c \sum_{i*}^{r-1} \left\{ \frac{n_{i*j} f_{i*j}}{n_{i*j} \alpha_{rc} + \alpha_e} \right\}$$

in the order given. The resulting system of equations consists of

$$f_{ij}: (\alpha_{rc} n_{ij} + \alpha_e) w_{iji'j'} - (\alpha_{rc} n_{rj} + \alpha_e) w_{rji'j'}$$

$$- (\alpha_{rc} n_{ic} + \alpha_e) w_{ici'j'} \qquad (3.5)$$

$$+ (\alpha_{rc} n_{rc} + \alpha_e) w_{rci'j'} = 0 \ ,$$

$$f_{ic}: (\alpha_{rc} n_{ic} + \alpha_e) p_i w_{ici'j'} - \sum_{j*} n_{ij*} w_{rj*i'j'}$$

$$-(\alpha_{rc} n_{rc} + \alpha_e) p_i w_{rci'j'} = \qquad (3.6)$$

$$= \alpha_r \left[n_{rj'} w r_{ri'j'} - n_{ij'} w r_{ii'j'} \right] \ ,$$

$$f_{rj}: (\alpha_{rc} n_{rj} + \alpha_e) q_j w_{rji'j'} - \sum_{i*} k_{i*j} w_{i*ci'j'}$$

$$-\alpha_{rc} n_{rc} + \alpha_e) q_j w_{rci'j'} \qquad (3.7)$$

$$= \alpha_c \left[n_{i'c} w c_{i'cj'} - n_{i'j} w c_{i'jj'} \right] \ ,$$

and

$$f_{rc}: \alpha_c \sum_{j*} n_{rj*} w_{rj*i'j'} + \alpha_c \sum_{i*} n_{i*c} w_{i*ci'j'}$$

$$+ (\alpha_{rc} n_{rc} + \alpha_e) w_{rci'j'} \qquad (3.8)$$

$$= - \alpha_r n_{rj'} w r_{ri'j'} - \alpha_c n_{i'c} w c_{i'cj'}$$

for $0 < i < r$ and $0 < j < c$.

Also

$$h_{ij} = \frac{\alpha_e \alpha_r (n_{rj} - n_{ij})}{\alpha_{rc} n_{ij} + \alpha_e}$$

and

$$k_{ij} = \frac{\alpha_e \alpha_c (n_{ic} - n_{ij})}{\alpha_{rc} n_{ij} + \alpha_e}$$

for all i,j.

The final two sets of equations (which are solved numerically) are obtained by combining the f_{ic} , i=1,···, r and f_{rj} , j=1,···, c equations. In the first case, they are combined to eliminate all unknowns but $w_{rji'j'}$ for j=1,···, c and in the second case, to eliminate all unknowns but $w_{ici'j'}$ for i=1,···, r.

The first set is obtained by forming the c equations

$$f_{rj} + \sum_{i}^{r-1} \left\{ \frac{k_{ij} f_{ic}}{(\alpha_{rc} n_{ic} + \alpha_e) p_i} \right\} \quad \text{for } j = 1, \cdots, c-1$$

and

$$f_{rc} - \alpha_c \sum_{i}^{r-1} \left\{ \frac{n_{ic} f_{ic}}{(\alpha_{rc} n_{ic} + \alpha_e) p_i} \right\} \quad .$$

The c equations which are solved numerically are

$$(\alpha_{rc} n_{rj} + \alpha_e) q_j w_{rji'j'}$$

$$- \sum_{j*}^{c} \sum_{i}^{r} \left[\frac{k_{ij} h_{ij*}}{(\alpha_{rc} n_{ic} + \alpha_e) p_i} \right] w_{rj*i'j'} \qquad (3.9a)$$

$$- (\alpha_{rc} n_{rc} + \alpha_e) q_c w_{rci'j'}$$

$$= \alpha_c \left[n_{i'c} w c_{i'cj'} - n_{i'j} w c_{i'jj'} \right]$$

$$+ \sum_{i}^{r} \left(\frac{\alpha_r k_{ij} \left[n_{rj'} w r_{ri'j'} - n_{ij'} w r_{ii'j'} \right]}{(\alpha_{rc} n_{ic} + \alpha_e) p_i} \right)$$

for $j = 1, \cdots, c-1$

and

$$(\alpha_{rc}n_{rc} + \alpha_e)q_c w_{rci'j'}$$

$$+ \sum_{j^*} \alpha_r n_{rj^*} + \alpha_c \sum_i \left(\frac{n_{ic}h_{ij^*}}{(\alpha_{rc}n_{ic} + \alpha_e)p_i}\right) w_{rj^*i'j'}$$

$$= -\alpha_r n_{rj'} w_{ri'j'} - \alpha_c n_{i'c} w_{ci'cj'} \qquad (3.9b)$$

$$- \sum_i^r \left(\frac{\alpha_r \alpha_c n_{rc}\left[n_{ij} w_{ri'j'} - n_{ij'} w_{ii'j'}\right]}{(\alpha_{rc} n_{ic} + \alpha_e)p_i} \right) .$$

The second system of equations, which yields $w_{ici'j'}$ for $i = 1, \cdots, r$ is obtained by similar manipulations. Combine

$$f_{ic} + \sum_j^{c-1} \left\{ \frac{h_{ij} f_{rj}}{(\alpha_{rc}n_{rj} + \alpha_e)q_j} \right\}$$

for $i = 1, \cdots, r-1$

and

$$f_{rc} - \alpha_r \sum_j^{c-1} \left\{ \frac{n_{rj} f_{rj}}{(\alpha_{rc}n_{rj} + \alpha_e)q_j} \right\} .$$

The resulting equations are

$$(\alpha_{rc}n_{ic} + \alpha_e)p_i w_{ici'j'}$$

$$- \sum_{i^*}^r \left[\sum_j^c \frac{h_{ij}k_{i^*j}}{(\alpha_{rc}n_{rj} + \alpha_e)q_j} \right] w_{i^*ci'j'}$$

$$- (\alpha_{rc}n_{rc} + \alpha_e)p_r w_{rci'j'}$$

$$= \alpha_r n_{rj'} w_{ri'j'} - n_{ij'} w_{ii'j'}$$

$$+ \sum_j^c \left(\frac{\alpha_c h_{ij}\left[n_{i'c} w_{ci'cj'} - n_{i'j} w_{ci'jj'}\right]}{(\alpha_{rc}n_{rj} + \alpha_e)q_j} \right)$$

for $i = 1, \cdots, r-1$ and

$$(\alpha_{rc}n_{rc} + \alpha_e)p_r w_{rci'j'}$$

$$+ \sum_{i^*} \alpha_c n_{i^*c} + \alpha_r \sum_j \left(\frac{n_{rj}k_{i^*j}}{(\alpha_{rc}n_{rj} + \alpha_e)q_j} \right) w_{i^*ci'j'}$$

$$= -\alpha_r n_{rj'} w_{ri'j'} - \alpha_c n_{i'c} w_{ci'cj'} \qquad (3.10b)$$

$$- \sum_j^c \left(\frac{\alpha_r \alpha_c n_{rj}\left[n_{i'c} w_{ci'cj'} - n_{i'j} w_{ci'jj'}\right]}{(\alpha_{rc}n_{rj} + \alpha_e)\, q_j} \right) .$$

Notice that there is a slight redundancy in the two systems in that the value for $w_{rci'j'}$ is obtained from both sets of equations. The remaining $(r-1)(c-1)$ values for specific i' and j' are available immediately via (3.5).

To obtain the complete set of r^2c^2 constants, note that there is much symmetry in the $\{w_{iji'j'}\}$. In particular

$$w_{iji'j'} = w_{i'jij'} = w_{ij'i'j} = w_{i'j'ij} .$$

It follows, that since only the right hand sides of (3.9a , 3.9b) and (3.10a, 3.10b) depend on i' and j' it follows that the system of r equations needs to be solved with r right hand sides i.e. for $i'=1,\cdots,r$ and fixed j' and the system of c equations needs to be solved with c right hand sides i.e. for $j'=1,\cdots,c$ and fixed i'. The remaining $\{w_{iji'j'}\}$ follow from (3.5).

4. VARIANCE COMPONENTS

The fact that the matrix X_0 consists of only a column of ones leads to a compact formula for Q_y. The element corresponding to row ijk and column i'j'k' follows immediately as

$$qv_{ijki'j'k'} = w_{iji'j'} + wc_{iji'j'}\delta_{ii'} + wr_{ii'j'}\delta_{jj'} + wrc_{ij}\delta_{ii'}\delta_{jj'} \qquad (4.1)$$

$$+ \delta_{ii'}\delta_{jj'}\delta_{kk'}/\alpha_e - m_{ij}m_{i'j'}/w$$

where $m_{ij} = \sum_{i'j'} w_{iji'j'} - (1-p_i^{-1}-q_j^{-1})/(\alpha_{rc}n_{ij} + \alpha_e)$

and $w = \sum\sum m_{ij}n_{ij}$. It will occasionally be convenient to write this as

$$qv_{iji'j'} + \delta_{ii'}\delta_{jj'}\delta_{kk'}/\alpha_e .$$

The final step is to obtain the system of equations.

$$\sum_j \operatorname{tr}(Q_y V_i Q_y V_j)\sigma_j^2 = Y' Q_y V_i Q_y Y \qquad (4.2)$$

where $j,i = r,c,rc,e$ and solve for σ_r^2, σ_c^2, σ_{rc}^2 and σ_e^2. Note that the reduced cases where it is assumed $\sigma_{rc}^2 = 0$ and where $\{n_{ij}\}$ are zero or one lead to systems with three equations. Expressions for the 14 distinct terms required in (4.2) are obtained directly and are available in the appendix.

In order to apply the technique prior values for the variance components σ_r^2, σ_c^2, σ_{rc}^2

and σ_e^2 (or at least their relative magnitudes) are required. In practice these are often not available and it is recommended that reasonable values be used to begin the procedure and the whole scheme iterated. Of course, negative values must be replaced by zeros. Experience indicates that convergence will be rapid.

If it is also assumed that the random components in the model have normal distributions and the iterative scheme proposed above converges then the solutions are exactly the restricted maximum likelihood estimates (Harville 1977) of the variance components. Also, the asymptotic variance-covariance matrix of the estimates is given by two times the inverse of the matrix on the left side of (4.1).

REFERENCES

Harville, D.A. (1977), Maximum Likelihood Approaches to Variance Component Estimation and to Related Problems, Journal of the American Statistical Association, 72, 320-338.

Hemmerle, William J., and Hartley, H.O. (1973), Computing Maximum Likelihood Estimates for the Mixed A.O.V. Model Using the W Transformation, Technometrics, 15, 819-831.

Lee, A.J. (1976), Estimation of Variance Components in Large Herd - by - Sire Designs with Interaction, Journal of Dairy Science, 59, 2138-2145.

Liu, Lon-Mu and Sentura, Jerome (1977), Computation of MINQUE Variance Component Estimates, Journal of the American Statistical Association, 72, 867-868.

Rao, C.R. (1971a), Estimation of Variance and Covariance Components - MINQUE Theory, Journal of Multivariate Analysis, 1, 257-275.

Rao, C.R. (1971b), Minimum Variance Quadratic Unbiased Estimation of Variance Components, Journal of Multivariate Analysis, 1, 445-465.

Rao, C.R. (1972), Estimation of Variance and Covariance Components in Linear Models, Journal of the American Statistical Association, 67, 112-115.

APPENDIX

Compact expressions for the 14 distinct elements of (4.2) follow directly from the expressions for elements of Q_V in (4.1) and the definitions of V_r, V_c, V_{rc} and V_e given by (2.3). Note in particular the definition of $\{qv_{ij,i'j'}\}$.

$$\operatorname{tr}(Q_V V_r Q_V V_r) = \sum_{ii'}\sum_{jj'}\Big(\sum_{ij} n_{ij} n_{i'j'} qv_{ij,i'j'}\Big)^2$$

$$+ 2\sum_{i} n_{i+} \sum_{jj'} \sum_{ij} n_{ij} n_{ij'} qv_{ij,ij'} \sigma_e^{-1} + \sum_{i} n_{i+}^2 \sigma_e^{-2} ,$$

where $n_{i+} = \sum_j n_{ij}$.

$$\operatorname{tr}(Q_V V_r Q_V V_c) = \sum_{ij}\sum_{i'j'}\Big(\sum n_{ij} n_{i'j'} qv_{ij,i'j'} + n_{ij}\sigma_e^{-1}\Big)^2 .$$

$$\operatorname{tr}(Q_V V_r Q_V V_{rc}) = \sum_{ij} n_{ij}^2 \sum_{i'j'}\Big(\sum n_{i'j'} qv_{ij,i'j'}\Big)^2$$

$$+ 2\sum_{ij} n_{ij}^2 \sum_{j'} n_{ij'} qv_{ij,ij'} \sigma_e^{-1} + \sum_{ij} n_{ij}^2 \sigma_e^{-2}$$

$$\operatorname{tr}(Q_V V_r Q_V) = \sum_{ij} n_{ij} \sum_{i'j'}\Big(\sum n_{i'j'} qv_{i'j',ij}\Big)^2$$

$$+ 2\sum_{ij} n_{ij} \sum_{j'} n_{ij'} qv_{ij',ij} \sigma_e^{-1} + n_{++}\sigma_e^{-2} ,$$

where $n_{++} = \sum\sum n_{ij}$.

$$\operatorname{tr}(Q_V V_c Q_V V_c) = \sum_{jj'}\Big(\sum_{ii'} n_{ij} n_{i'j'} qv_{ij,i'j'}\Big)^2$$

$$+ 2\sum_{j} n_{+j} \sum_{ii'} n_{ij} n_{i'j} qv_{ij,i'j} \sigma_e^{-1} + \sum_{j} n_{+j}^2 \sigma_e^{-2} ,$$

where $n_{+j} = \sum_i n_{ij}$.

$$\operatorname{tr}(Q_V V_c Q_V V_{rc}) = \sum_{ij} n_{ij}^2 \sum_{i'j'}\Big(\sum_{i} n_{i'j'} qv_{ij,i'j'}\Big)^2$$

$$+ 2\sum_{ij} n_{ij}^2 \sum_{i'} n_{i'j} qv_{ij,i'j} \sigma_e^{-1} + \sum_{ij} n_{ij}^2 \sigma_e^{-2} .$$

$$\operatorname{tr}(Q_V V_c Q_V) = \sum_{ij} n_{ij} \sum_{i'j'}\Big(\sum_{i'} n_{i'j'} qv_{i'j',ij}\Big)^2$$

$$+ 2\sum_{ij} n_{ij} \sum_{i'} n_{i'j} qv_{i'j,ij} + n_{++}\sigma_e^{-2} .$$

$$\operatorname{tr}(Q_V V_{rc} Q_V V_{rc}) = \sum_{ij} n_{ij}^2 \sum_{i'j'} n_{i'j'}^2 qv_{ij,i'j'}^2$$

$$+ 2\sum_{ij} n_{ij}^3 qv_{ij,ij} \sigma_e^{-1} + \sum_{ij} n_{ij}^2 \sigma_e^{-2} .$$

$$\operatorname{tr}(Q_V V_{rc} Q_V) = \sum_{ij} n_{ij} \sum_{i'j'} n_{i'j'}^2 qv_{ij,i'j'}^2$$

$$+ 2\sum_{ij} n_{ij}^2 qv_{ij,ij} \sigma_e^{-1} + n_{++}\sigma_e^{-2} .$$

$$\mathrm{tr}(\underline{Q}_v\underline{Q}_v) = \sum_{ij}\sum n_{ij}\sum_{i'j'}\sum_{i'j'} n_{i'j'}\, qv^2_{iji'j'}$$

$$+ 2\sum_{ij}\sum n_{ij}\, qv_{ijij}a_e^{-1} + n_{++}a_e^{-2}$$

$$\underline{Y}'\underline{Q}_v\underline{V}_r\underline{Q}_v\underline{Y} = \sum_i\left(\sum_{i'j'}\sum y_{i'j'+} + \sum_j n_{ij}\, qv_{iji'j'+} y_{i++}a_e^{-1}\right)^2,$$

where $y_{ij+} = \sum_k y_{ijk}$ and $y_{i++} = \sum_j y_{ij+}$.

$$\underline{Y}'\underline{Q}_v\underline{V}_c\underline{Q}_v\underline{Y} = \sum_j\left(\sum_{i'j'}\sum y_{i'j'+} + \sum_i n_{ij}\, qv_{iji'j'+} y_{+j+}a_e^{-1}\right)^2,$$

where $y_{+j+} = \sum_i y_{ij+}$.

$$\underline{Y}'\underline{Q}_v\underline{V}_{rc}\underline{Q}_v\underline{Y} = \sum_{ij}\sum\left(n_{ij}\sum_{i'j'}\sum qv_{iji'j'}y_{i'j'+} + y_{ij+}a_e^{-1}\right)^2.$$

$$\underline{Y}'\underline{Q}_v\underline{Q}_v\underline{Y} = \sum_{ij}\sum\left\{n_{ij}\left(\sum_{i'j'}\sum qv_{iji'j'}y_{i'j'+}\right)^2\right.$$

$$\left. + 2y_{ij+}\sum_{i'j'}\sum qv_{iji'j'}y_{i'j'+}a_e^{-1} + \sum_k y^2_{ijk}a_e^{-2}\right\} .$$

Analysis of Data From A Research Aircraft

J. E. GRIMES
California Polytechnic State University
&
NASA-Ames Research Center

During the flight test portion of the testing program of a research aircraft, various sensor devices, such as strain gages, are placed at various points on the aircraft to measure the variation of critical parameters. The data must be analyzed in near-real-time as well as off-line in order to ensure the safety and efficiency of the flight test program. This paper gives a general overview of a computer system and the associated software that can be used to perform the required data analysis.

1.0 INTRODUCTION

During the flight of a research aircraft, data sampled from various points (strain gages, etc. are used as measuring devices) on the aircraft are monitored on the ground by research engineers. For safety of flight, continous, uninterrupted voice and data signals originating in the research aircraft must be available for the monitoring research engineers while the aircraft is airborne. When NASA-Ames Research Center tests an aircraft, such as the Tilt Rotor Research Aircraft (Figure 1), it is desirable to do some dynamic analysis computations in a near real-time mode so as to improve the efficiency of the testing and assure greater safety. A computer system, along with the appropriate software, must be available to perform the tasks of data acquisition, processing, and reduction. Because the data cannot always be completley analyzed in a near real-time mode, it must be collected on tape or disc so that it can be considered further after the completion of the flight. This paper looks at the approach that NASA-Ames is implementing to handle the problem of data collection and dynamic analysis.

2.0 FLIGHT DATA ACQUISITION AND ANALYSIS SYSTEM

In general, at NASA-Ames Research Center data from the research aircraft is transmitted by means of a pulse code modulated (PCM) data stream to a ground station where the task of acquisition is completed and analysis is performed. Because the aircraft testing is to be performed at two sites separated by mountain ranges (See Figure 2), real-time data acquisition systems must exist at both, with a data communication link between the sites so that the complete data analysis system is necessary at only one site.

The completed data acquisition and analysis system, necessary to meet flight requirements at NASA-Ames, can be satisfied by any one of several system architectures. The approach to be used is an approach standardized for use in the NASA-Ames wind tunnel facilities. The architecture for the Flight Data System is a distributed system approach using parallel data

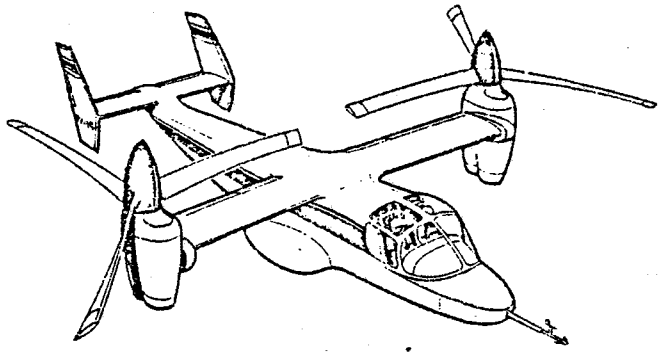


FIGURE 1

SOLUTION TO FLIGHT TEST DATA SYSTEM REQUIREMENTS

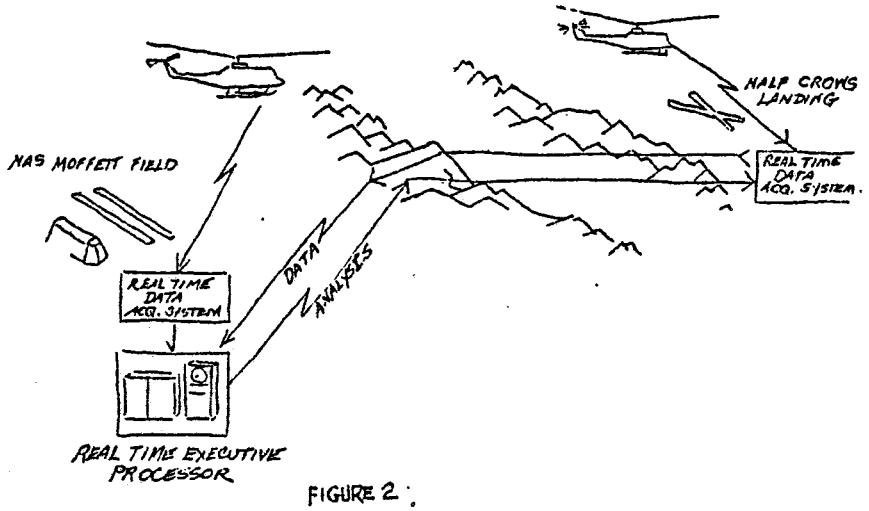
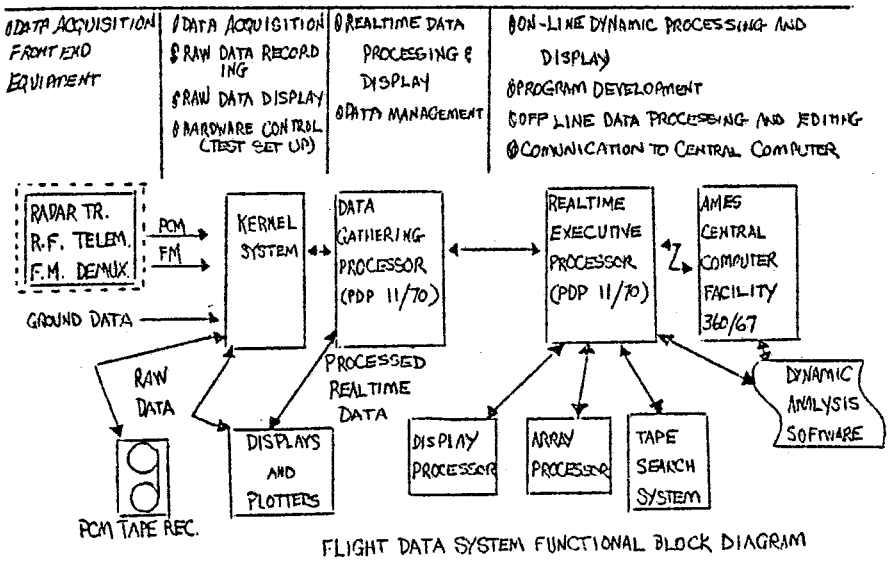


FIGURE 2



FLIGHT DATA SYSTEM FUNCTIONAL BLOCK DIAGRAM

FIGURE 3

and processing paths when and where high data and processing rates are required (Figure 3). The system is partitioned into functicnal blocks in a manner so that parallel modules can be added for increased capacity or more powerful modules can be added for more sophisticated capability, and the three blocks are the:

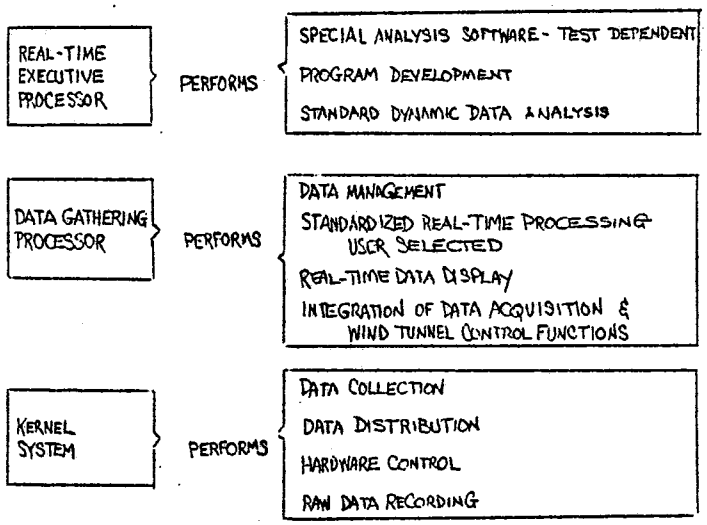
a) Kernel System
b) Data Gathering Processor, and
c) Real-time Executive Processor.

The generalized function of each block is given in Figures 3 and 4.

The detailed breakdown of each of the blocks is given in Figures 5, 6, and 7. It is not the intent or within the scope of this paper to look at these components in detail.

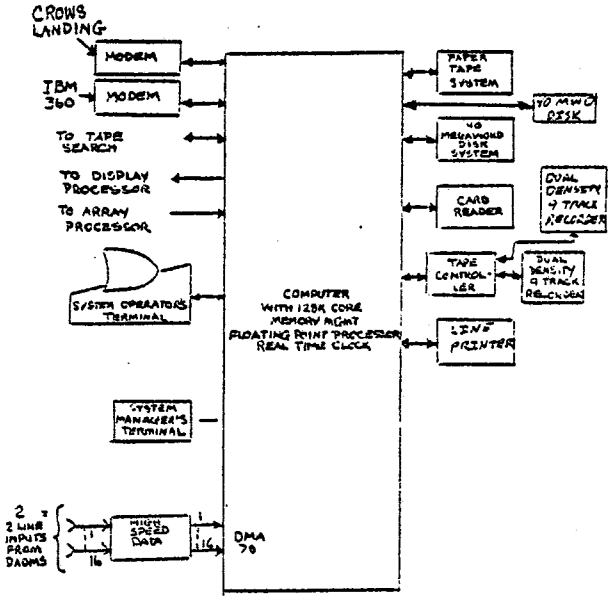
This Flight Data System has capability for:

a. Decomutation of PCM data streams.
b. PCM recording on analog magnetic tape.
c. Digitizing and recording data from ground support.
d. Driving strip chart recorders.
e. Bar chart display of up to 128 data channels.
f. Display of raw data CRT terminal.
g. Data management.
h. Inserting dummy words in data streams if signal is lost.
i. Computating and displaying various other quick look information.
j. Interactively performing dynamic analyses in a near real-time environment.



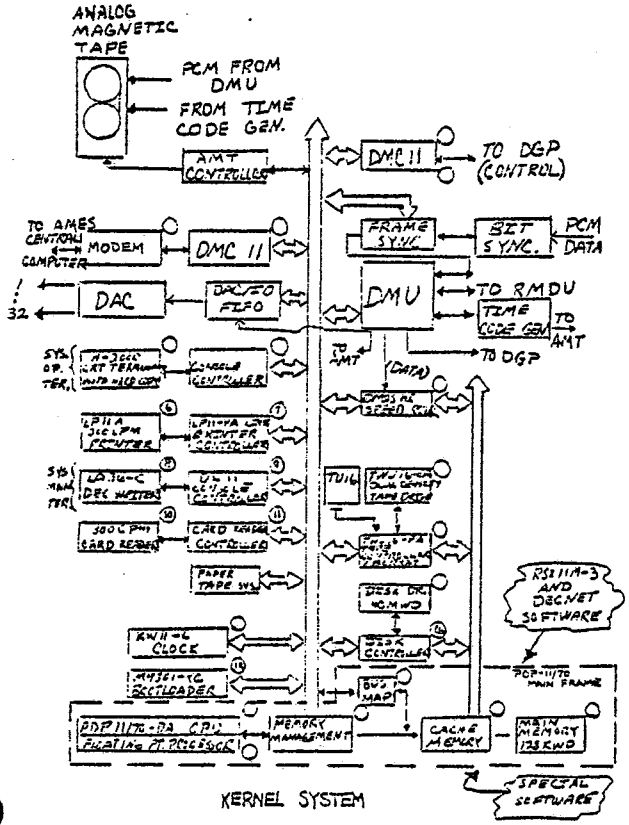
SUBSYSTEM FUNCTIONS

FIGURE 4



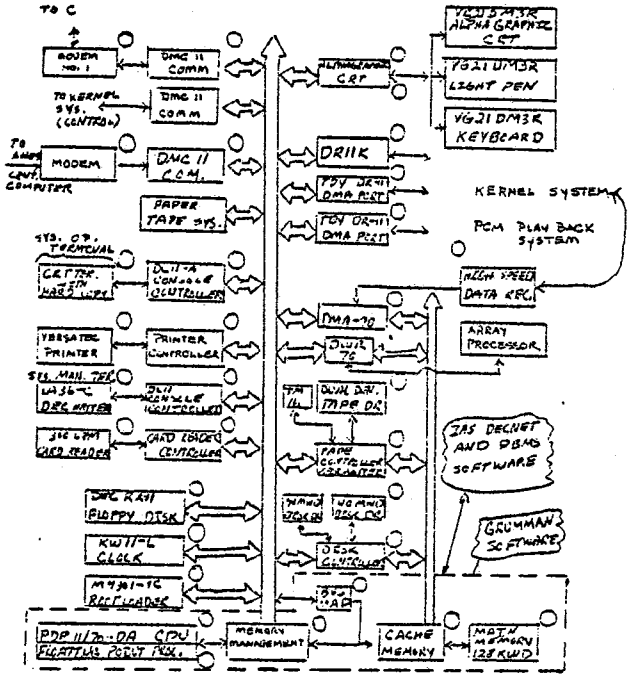
DATA GATHERING PROCESSOR

FIGURE 6



KERNEL SYSTEM

FIGURE 5



REAL TIME EXECUTIVE PROCESSOR SYSTEM

FIGURE 7

3.0 DYNAMIC ANALYSIS

In determining the type of software to be used in dynamic analysis the following items were considered:

A. Time Factor. Flight testing schedules for the Tilt Rotor Aircraft allowed only one year for implementation.

B. Expandability. The software must be such that it can be easily expanded to handle data from various aircraft in various environments.

C. Portable. Initially the software was to reside on the IBM 360/70 and then at a later date it was to be transferred to a combination PDP 11/70 and array processor. Hence as much of the coding as possible should be in FORTRAN.

D. Accuracy. The analysis algorithms should be capable of accurate analysis in difficult environments.

Various alternatives were considered and it was determined that an off-the-shelf package would be used. The package was developed and used by Grumman Data Systems Corporation.

A block diagram of the system is given in Figure 8 and a functional system software diagram is given in Figure 9. Features of the software package include:

A. Minimal software development risk since it was fully developed.

B. Fully expandable, expecially in terms of applications modules.

C. Highly portable because initially most of the coding was to be ANSI FORTRAN, with primarily only the IBM utility routines coded in assembly language.

The capabilities of the applications modules are of prime interest to the aircraft research engineer, and the ability to add new application modules is of high importance.

The application module consists of three sub-modules - INAME, RNAME, and ENAME. Each of the original application modules, as well as any added application modules, must be composed of sub-modules. The sub-module's purposes are as follows:

A. INAME Module

The initialization module for an applications program called NAME (e.g. Name might be TLEFAD, APSD, etc.). The INAME module is stored on an application program file along with the related RNAME and ENAME modules.

B. RNAME Module

The data analysis module for an applications program called NAME.

C. ENAME Module

The optional program termnation module for an applications program called NAME. This module will prepare the results of the computations for output.

Included with the original software package from Grumman Data Systems Corporation were the six applications modules TLEFAD, GIFMAP, APSD, FORIER, TIMHST, and FILHST. The functions of the application modules are:

A. TLEFAD. This program was specially designed to track the migration of modal resonant frequencies and damping coefficients as the flight envelope of an aircraft is expanded. The utilization of this program requires that the user have some knowledge of the modal composition of the flutter response data; this information is obtained from previous structural analyses, ground vibration surveys, test results, etc. If the modal composition is not available, it can be readily obtained from an initial frequency domain analysis of the data via the GIFMAP, APSD, or FORIER programs or from time history plots and/or strip chart records. Filtered and unfiltered data channel time history plots are available from the FILHST and TIMHST programs, respectively.

The utilization of the TLEFAD program is particularly desirable when timely decisions on aircraft flight test envelope expansion must be made, since speed of computation, flexibility, and noise rejection are improved by use of the known modal information. In addition, cross-checking by analysis of data from independent response transducers may be accomplished simultaneously, enhancing user confidence in the resonant frequency and damping results. The analytical techniques employed are capable of processing flutter response data with or without a measured driving function signal and are compatible with, and in no way limited to, any one of the following means of structural excitation:

- o swept frequency excitation,
- o random excitation,
- o abrupt control surface inputs,
- o shake and stop excitation
- o impulsive input excitation.

Basically, the TLEFAD program uses a least-squares z-transform/difference-equation error modeling method in conjunction with correlation and digital filtering techniques (see two previously quoted references for theoretical details) to determine the modal resonant frequencies and damping coefficients associated with the aeroelastic response of the aircraft structure.

The primary output of the program consists of a tabulation of the resonant frequency and damping coefficient results obtained for each modeled mode in every data segment. These results are augmented by diagnostic information if real poles are detected or if difficulties are encountered in extracting all the roots of the specified difference-equation model. Auxiliary

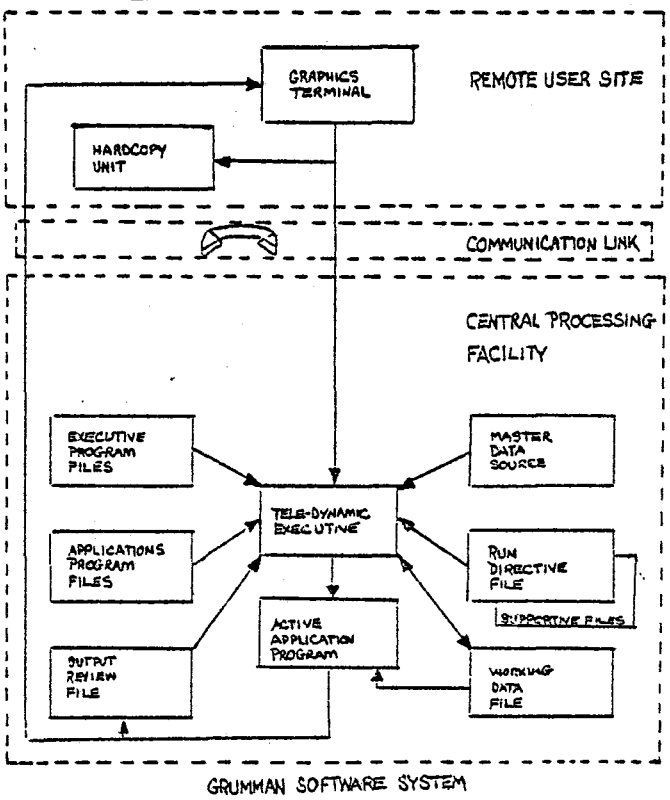


FIGURE 8

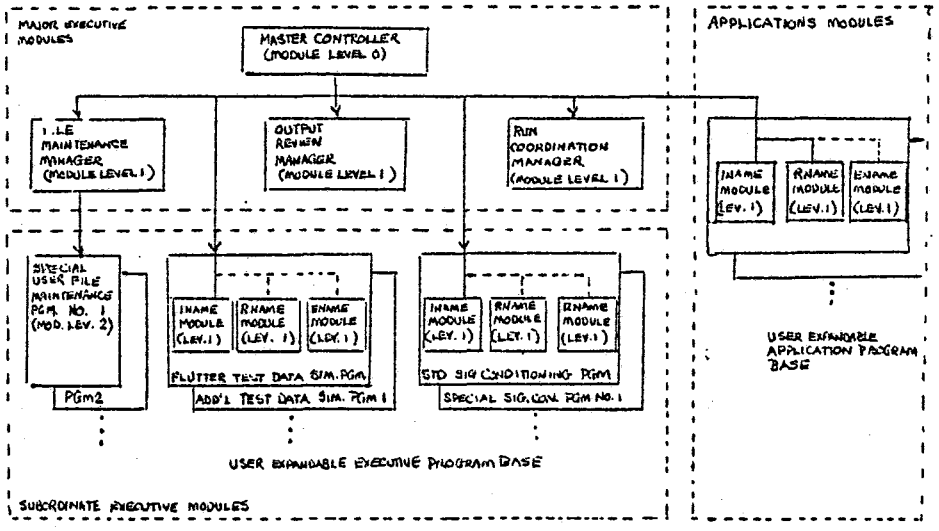
information defining the general quality of the data fit and a reference to aircraft altitude/airspeed (if such data measurements are provided) for each calculated set of results are also included in this primary output tabulation.

Secondary outputs of the program include a tabulation of backup information, which is helpful in assessing the validity of results, a plot of calculated shaken frequency versus time when swept frequency data as analyzed and plots of any computed correlation functions.

B. GIFMAP. This program is primarily designed to evaluate swept frequency or random structural response data to determine if any significant modes of vibration have been excited. Calculated signal frequency response functions provide the basis for determining modal content and for extracting estimates for corresponding resonant frequency and damping coefficient information. GIFMAP is particularly useful in analyzing data segments whose modal content is either unknown or uncertain.

If the test data contains a system driving function measurement, GIFMAP can be directed to compute the cross-correlation function between system input/response quantities and the autocorrelation function of the system input. By transforming the resulting correlation information into the frequency domain and dividing the resulting cross-spectrum by the autospectrum, GIFMAP computes a frequency response function respresenting the transfer function characteristics of the system under test. On the other hand, if the nature of the test data is consistent with the requirements of autocorrelation or random decrement signature analysis, the program can compute frequency response information through the transformation of either one of these two functions. Although the frequency response functions computed from an autocorrelation function or a random decrement signature are somewhat different in form, they both can be considered representative of a transfer function characteristic possessing poles identical to the actual system under test.

System resonant frequency and damping coefficient can be determined by means of the frequency response component analysis method or by the least-squares z-transform/difference-equation error modeling method. The first method is only useful where sufficient modal frequency separation exists. The second method provides the primary means by which GIFMAP reduces frequency response information to determine the overall modal characteristics of



FUNCTIONAL DIAGRAM OF GRUMMAN SOFTWARE SYSTEM
FIGURE 9

the data. In this case frequency response function information is windowed and transformed back into the time domain for analysis via the least-squares difference-equation identification algorithm. The user can select the manner in which the frequency response function will be windowed and modeled or allow the program to establish analysis procedure in an automatic fashion. In the interactive real-time operating mode the user has the ability to override automatic windowing and modeling selections made by the program.

Primary GIFMAP outputs consist of tabulation of calculated resonant frequency and damping coefficient information, as well as plots of the computed frequency response function and its real and imaginary components. Optional secondary program outputs consist of plots of calculated autocorrelation functions, random decrement signatures, or mathematically reconstructed frequency response function information where appropriate.

C. APSD. The APSD program can compute and output plots for signal power spectral density, power spectral distribution amplitude spectrum, and autocorrelation function information for multiple channel input data. Signal power peak tabulations are also available. The specific outputs to be calculated in a program run are user selectable. Data reduction can be implemented by either the transform method using the Fast Fourier Transform (FFT) algorithm or the classical autocorrelation calculation technique. In either case, Bartlett, Hanning, or Hamming data weighting options can be selected for smoothing results.

Results are calculated and averaged over the total interval of time necessary to achieve the statistical accuracy implied by the user specified degrees of freedom. APSD can be directed to output accumulated intermediate results in fixed incremental degree-of-freedom steps. This provides the user with a visual means of assessing the stability and convergency of power spectral estimates. The program can also be directed to output nonaccumulated, snapshot, results in user specified fixed incremental degrees-of-freedom steps which are helpful in establishing whether the data is stationary.

The ASPD program has wide application in the analysis of random vibration time series test data. In flutter data analysis it is often called upon to provide power spectral density plots for a given set of response signals. Its primary purpose is to provide a quick look at the overall vibrational energy distribution as a function of frequency. This information provides insight into the modal complexity of the data. Although APSD is not normally used to establish modal damping coefficient information, it can be estimated from a power spectral density plot using the one-half power method if the system input has a spectrum that is broad-band-flat.

D. FORIER. This program individually calculates the Discrete Fourier Transform corresponding to multiple channel data sets using the Fast Fourier Transform (FFT) algotithm. Output is presented in terms of a finite Fourier Series coefficient expansion on a fixed segment of each input data signal. The primary output of the FORIER program is a plot of total signal amplitude spectrum. However, the user has the option to select additional plot output corresponding to signal phase angle and/or its sin and cos amplitude spectrum components.

The FORIER program is particularly useful in the analysis of periodic signals such as those typically encountered in helicopter and propeller aircraft structural response data. Since, in these cases, it is not necessary to accumulate or average data over relatively long intervals for the purpose of achieving statistical accuracy, the response of the program is particularly quick. In flutter test data analysis a user can employ this program to establish the relative magnitude of blade reflection effects and modal response amplitudes. This information, in turn, can be used to establish proper modeling selections for subsequent analysis via the TLEFAD program.

E. TIMHST. The TIMHST program outputs time history plots for multiple channel user selected input signals. In vibration and flutter analysis these plots are primarily used to assess general signal level and quality.

F. FILHST. This program is similar to the TIMHST program with the exception that it outputs filtered time history plots. Filtering is accomplished by recursive digital filters with low-pass or band-pass Bessel or Butterworth characteristics. For each input signal the user can select the filter type, corner frequencies, and roll-off characteristics (up to 12th order) required.

It is anticipated that additional application modules will be added in the future when difficult testing environments arise, such that existing modules do not give accurate results.

CONCLUDING REMARKS

A general overview of the computer system, along with the dynamic analysis software, has been given in this paper. It is anticipated that because of the block design of the computer system and the expandability of dynamic analysis software the system will be efficient for many years in the future.

Variance Estimation Using Half-Sample Replications

by

Robert S. Jewett
Computer Systems Analyst
Bureau of the Census

This paper explains how the Bureau of the Census applies the balanced half-sample replication technique to calculate variance estimates for demographic surveys having a stratified sample design. Computer storage limitations make it difficult to use this technique if there are a large number of strata, and a computer program is presented that was developed to overcome this problem. A new proof is given to show the equivalence of the Keyfitz method and the balanced half-sample replication technique when estimating variances for simple linear statistics.

Introduction

At the Bureau of Census we often use the Balanced Half-Sample Replication (BHSR) technique to calculate variance estimates for surveys having stratified sample designs. In the past this has been done for the Longitudinal Manpower Survey, and it is presently being done for the National Travel Survey and for certain supplements to the National Crime Survey.

This method has several advantages in that it is easy to understand and to program, and it also produces reliable variance estimates. One drawback to the BHSR technique is that it requires the usage of an orthogonal matrix, and if the matrix is large, computer storage limitations can present a problem. This paper describes a computer program we have developed that makes it easier to overcome this difficulty when it arises. In addition, appendix A gives a new proof of the equivalence of the Keyfitz and BHSR variance estimates when applied to a simple linear estimate.

The BHSR Technique

The next few paragraphs describe how the BHSR technique may be used to calculate variance estimates for a statistic Y. Assume that the sample is divided into N independent strata, and that for each stratum s there are two independent half-sample estimates X_1^s and X_{-1}^s with the following properties:

1) the expected value of X_1^s = the expected value of X_{-1}^s.

2) the estimate X^s for the stratum = $X_1^s + X_{-1}^s$

3) The final estimate for Y is the statistic: $Y = \sum_{s=1}^{N} (X_1^s + X_{-1}^s)$

If we select one half-sample from each stratum, add them up, and multiply the result by 2, we will have an unbiased "replication estimate" for Y. There are 2^N possible different patterns for choosing half-samples from the strata, and if we tried each pattern and obtained 2^N unbiased replication estimates of Y, we could calculate the variance of this set of 2^N numbers and the result would be an estimate for the sampling variance of Y.

Of course, if N is large it becomes impractical, to calculate 2^N separate estimates. In this case, an orthogonal matrix can be used to define a much smaller set of replication estimates that can be used to estimate the variance. An orthogonal matrix is defined to have the following properties:

1) The size of the matrix is 2^n by 2^n, where n is a positive integer. The value for n must be chosen so that the number of strata is less than or equal to 2^n.

2) The elements of the matrix are +1 and -1.

3) The matrix multiplied by its transpose gives a matrix with the number 2^n along the diagonal, and zero elsewhere.

For example when n = 2 the 4 by 4 matrix is:

$$\begin{pmatrix} 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ -1 & -1 & -1 & 1 \end{pmatrix}$$

Given an orthogonal matrix with at least N columns, each row of the matrix can be used to determine a pattern for selecting half-samples from the strata, and a corresponding replication estimate may be calculated. When this is done for each row, we can compute the variance of this set of replication estimates about the overall estimate, and the result as McCarthy [5] shows would be the same as if all 2^N possible replications had been used. The exact formula to calculate the BHSR variance estimate is given in Appendix A.

The Keyfitz Alternative

In the case described above, the Keyfitz variance estimation technique [4] could also be used. The variance estimate would be

$$\sum_{s=1}^{N} (X_1^s - X_{-1}^s)^2$$

which would be exactly the same value as the replication variance estimate (See appendix A for a proof of this statement). This is obviously a simple calculation to perform, but a problem arises when we attempt to use the Keyfitz technique to estimate the variance of more complex statistics such as ratio estimates or regression coefficients. In order to use the Keyfitz technique, the estimation formula for such statistics must be linearized using a Taylor's series, and from our experience the resulting variance estimation procedure is often complicated and hard to program. On the other hand, the BHSR method may still be applied with only slightly more difficulty than in the case of the simple linear estimate.

Generating Orthogonal Matrices

As was mentioned before, one problem in using an orthogonal matrix is that it can require a great deal of computer storage. For example, if the matrix is of the size 256 by 256, then 65536 words are required to store the matrix, and under the present computer system at the Bureau of the Census a single program is not allowed to use this much storage.

We have now developed a way to create orthogonal matrices having the property that the entire matrix can easily be generated from the first column. This means that only one column of the matrix has to be resident in the computer, which solves the storage problem. Of course, it does take a certain amount of computer time to recalculate an element of the matrix each time it has to be used, but the calculation is so simple that it should not cost much.

The method we used to develop the matrix is taken directly from the paper Constructing Orthogonal Replications for Variance Estimation. [2], which tells how to generate an orthogonal matrix using two parameters p and n, where p is a prime number and n is any positive integer. Using the value p = 2, a computer program was written that follows the procedure described in the paper and creates an orthogonal matrix for any value of n up to 12.

The following is a general description of the steps to create a 2^n by 2^n orthogonal matrix.

Step 1: Consider the ring of polynomials that have coefficients in the field of integers modulo 2. Find an irreducible polynomial of degree n.

Example: $X^3 + X+1$ is irreducible.

Step 2: The ring of polynomials modulo the irreducible polynomial forms a Galois field having 2^n elements.

A generator of the field is a polynomial $p(x)$ such that $p(x)^k=1$ if and only if $k=2^n-1$

Find a generator for the field.

Example: a generator is X + 1.

Step 3: Write the non-zero elements of the field as powers of the generator.

Example:

$(X + 1)^1 = X + 1$ $\qquad$ $(X + 1)^5 = X$

$(X + 1)^2 = X^2 + 1$ $\qquad$ $(X + 1)^6 = X^2 + X$

$(X + 1)^3 = X^2$ $\qquad$ $(X + 1)^7 = 1$

$(X + 1)^4 = X^2 + X + 1$

Step 4: In each element that is listed, the coefficient in the units place is equal to either zero or one. Form a vector of length 2^n-1 consisting of these coefficients in the same order they appear in the list.

Example: The vector of length 7 is (1,1,0,1,0,0,1).

This vector is the one that is created in the computer program. The program is designed to read in any value of n up to 12, and to then calculate the appropriate vector of length 2^n-1. The next two steps show how the vector can be used to form an orthogonal matrix.

Step 5: Using this vector as the first column of a matrix, form each succeeding column by moving every element of the original column up one place, and moving the topmost element to the bottom. This matrix is of the size 2^n-1 by 2^n-1.

Example:

1	1	0	1	0	0	1
1	0	1	0	0	1	1
0	1	0	0	1	1	1
1	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	1	1	0	1	0
1	1	1	0	1	0	0

Step 6: Change all zeros in the matrix to -1. Add on a row of -1, and make an additional column with every element equal to one. This forms an orthogonal matrix of size 2^n by 2^n.

1	1	-1	1	-1	-1	1	1
1	-1	1	-1	-1	1	1	1
-1	1	-1	-1	1	1	1	1
1	-1	-1	1	1	1	-1	1
-1	-1	1	1	1	-1	1	1
-1	1	1	1	-1	1	-1	1
1	1	1	-1	1	-1	-1	1
-1	-1	-1	-1	-1	-1	-1	1

From the last two steps it is easy to see that if the vector of length 2^n-1 is stored in the main memory of the computer, it is easy to calculate any other element of the matrix. In the past, we had only used this method of constructing orthogonal matrices for the case n=7, but we are now using matrices with n=8 and n=9 to obtain variance estimates. Here are some of the vectors that are calculated for certain values of n.

n = 3, the vector is 1 1 0 1 0 0 1

n = 4, the vector is 1 1 1 0 0 0 1 0 0 1 1 0 1 0 1

n = 5, the vector is 1 1 1 1 0 1 0 0 0 1 0 0 1 0 1 0
1 1 0 0 0 0 1 1 1 0 0 1 1 0 1

The computer program that does the preceding steps is listed in Appendix B.

APPENDIX A

Assume that the estimate X is a simple sum of other estimates which are taken from independent strata. That is, each stratum was subsampled independently from all the others, and

$$Y = \sum_{n=1}^{N} X_n \quad ,$$

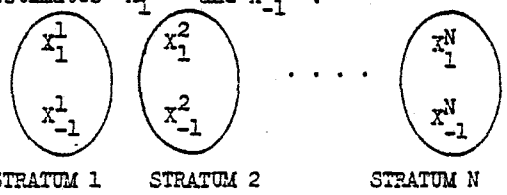
where X_n is the estimate for stratum n. The purpose of this appendix is to prove that the Keyfitz variance estimate of Y is equal to the balanced half-sample replication variance estimate.

Preliminaries:

The standard Euclidean norm of a vector (denoted as $||x||$) is defined to be the square root of the inner product of the vector with itself.

If A is an N by N orthogonal matrix, we can use the property that the matrix multiplied by its transpose equals the identity matrix multiplied by N to easily show that $||Ax|| = \sqrt{N}||x||$, where x is an N-dimensional vector. Note that N is a power of 2.

Assume that the sample is divided into N strata, and that for each stratum j there are two half-sample estimates X_1^j and X_{-1}^j.



Define these two vectors of dimension N:

$$Y_1 = (X_1^1 \ , X_1^2 \ , X_1^3 \ , \ldots X_1^N)$$

$$Y_{-1} = (X_{-1}^1 \ , X_{-1}^2 \ , X_{-1}^3 \ , \ldots X_{-1}^N)$$

The BHSR Variance Estimate:

Any row of the orthogonal matrix A can be used to define a replication estimate by letting each element of the row refer to one of the two half-samples from the corresponding stratum. For example, if the j^{th} element of the row is equal to -1, then we choose half-sample X_{-1}^j from the j^{th} stratum. Likewise, if the k^{th} element is +1, then the half-sample X_1^k is selected from stratum k. Adding the half samples selected and multiplying by 2, we obtain the replication estimate for the row.

Using the elements of the orthogonal matrix as subscripts, we can define the replication estimate for any row i.

$$\text{Rep. Est.}_i = 2\sum_{j=1}^{N} X^j_{A(i,j)}$$

The BHSR variance estimate is equal to

$$\frac{1}{N}\sum_{i=1}^{N}(\text{Rep. Est.}_i - \text{overall estimate})^2 =$$

$$\frac{1}{N}\sum_{i=1}^{N}\left[2\sum_{j=1}^{N} X_{A(i,j)} - \sum_{j=1}^{N}(X^j_1 + X^j_{-1})\right]^2$$

The Keyfitz Variance Estimate:

The Keyfitz variance estimate is

$$\sum_{j=1}^{N}(X^j_1 - X^j_{-1})^2 = ||Y_1 - Y_{-1}||^2$$

LEMMA:

Show that for all i and j less than N

$$A(i,j)(X^j_1 - X^j_{-1}) = 2X^j_{A(i,j)} - (X^j_1 + X^j_{-1})$$

Proof: Case 1

If $A(i,j) = 1$, then

$$A(i,j)(X^j_1 - X^j_{-1}) = X^j_1 - X^j_{-1} = 2X^j_1 - X^j_1 - X^j_{-1}$$

$$= 2X^j_{A(i,j)} - (X^j_1 + X^j_{-1})$$

Proof: Case 2

If $A(i,j) = -1$, then

$$A(i,j)(X^j_1 - X^j_{-1}) = -X^j_1 + X^j_{-1} = 2X^j_{-1} - X^j_1 - X^j_{-1}$$

$$= 2X^j_{A(i,j)} - (X^j_1 + X^j_{-1})$$

Main Proof:

We now have that the BHSR variance estimate equals:

$$\frac{1}{N}\sum_{i=1}^{N}\left[2\sum_{j=1}^{N} X^j_{A(i,j)} - (X^j_1 + X^j_{-1})\right]^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{j=1}^{N}\left(2X^j_{A(i,j)} - (X^j_1 + X^j_{-1})\right)\right]^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{j=1}^{N} A(i,j)(X^j_1 - X^j_{-1})\right]^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left[\text{the } i^{th} \text{ element of the vector } A(Y_1 - Y_{-1})\right]^2$$

$$= \frac{1}{N}||A(Y_1 - Y_{-1})||^2 = \frac{1}{N}\left[\sqrt{N}||Y_1 - Y_{-1}||\right]^2$$

$$||Y_1 - Y_{-1}||^2 = \text{the Keyfitz variance estimate.}$$

Epilogue:

It was mentioned earlier that N must be a power of two. If we have a collection of M strata where M is not a power of two, we then choose N to be a power of two larger than M and create an orthogonal matrix of that size. The first M columns of this matrix can be used to define the replications. This can also be interpreted as creating (N-M) imaginary strata with half-sample estimates equal to zero.

In any event, with this set of N replications defined over the M strata, the BHSR variance estimate can be defined to be

$$\frac{1}{N}\sum_{i=1}^{N}(\text{Rep. Est.}_i - \text{overall estimate})^2 =$$

$$\frac{1}{N}\sum_{i=1}^{N}\left[\sum_{j=1}^{M} X^j_{A(i,j)} - \sum_{j=1}^{M}(X^j_1 + X^j_{-1})\right]^2$$

which is equal to the Keyfitz variance estimate

$$\sum_{j=1}^{M}(X^j_1 - X^j_{-1})^2.$$

APPENDIX B

```
C     THIS PROGRAM WAS WRITTEN BY ROBERT JEWETT, STATISTICAL METHODS
C     DIVISION, BUREAU OF CENSUS, SUITLAND, MARYLAND. TEL 301-763-2306.
C
C     THIS PROGRAM IS DESIGNED TO RUN ON A UNIVAC 1110 SERIES COMPUTER
C     WITH A STANDARD FORTRAN V COMPILER.  IF YOU WANT TO CONVERT THIS
C     PROGRAM TO RUN ON ANOTHER MACHINE, YOU WILL HAVE TO MODIFY THE
C     SUBROUTINE CREATE.
C
C

      INTEGER  P(30),Q(30),R(30),T(30),MODULO(30,15),SOLN
      INTEGER  S(30),COL(5000),SUM,A
      INTEGER  A2,APLUS1,AMIN1
C
C
C     THE PROGRAM READS IN THE VALUE FOR A, AND GENERATES THE FIRST COLUMN
C     OF THE 2**A BY 2**A ORTHOGONAL MATRIX.  THE WAY THE PROGRAM IS SET UP,
C     THE VALUE FOR A SHOULD BE LESS THAN OR EQUAL TO 12.
C
      READ(5,50) A
50    FORMAT()
      WRITE(6,60) A
60    FORMAT('1 A= ',I5///)
      APLUS1=A+1
      AMIN1=A-1
      A2=2**A-1
C
C
C     THIS PART OF THE PROGRAM FINDS AN IRREDUCIBLE POLYNOMIAL P(X)
C     OF DEGREE A.  THIS POLYNOMIAL IS STORED IN THE ARRAY P SUCH THAT
C     P(I) = THE COEFFICIENT OF THE X**(I-1) TERM.  IN FACT, THIS IS HOW
C     ALL POLYNOMIALS ARE STORED IN THE PROGRAM.
C
      NUM=A2+1
100   NUM=NUM+1
      IF (NUM.GT.2**A2-1) STOP
      CALL CREATE(P,NUM)
      CALL SOLVE(P)
      IF (SOLN.EQ.'YES') GO TO 100
      LIM=A/2
      DO 200 N=1,LIM
      LIM1=2**N+1
      LIM2=2**(N+1)-1
      DO 175 NUM1=LIM1,LIM2
      CALL CREATE(Q,NUM1)
      CALL SOLVE(Q)
      IF (SOLN.EQ.'YES') GO TO 175
      LIM3=2**(A-N)+1
      LIM4=2**(A-N+1)-1
      DO 150 NUM2=LIM3,LIM4
      CALL CREATE(R,NUM2)
      CALL SOLVE(R)
      IF (SOLN.EQ.'YES') GO TO 150
      CALL MULT(Q,R,T)

      DO 125 I=1,APLUS1
      IF (T(I).NE.P(I)) GO TO 150
125   CONTINUE
      WRITE(6,130) Q,R,P
130   FORMAT(' P IS NOT PRIME! Q=R=P'/' Q=',30I2/' R=',30I2/' P=',30I2/)
      GO TO 100
150   CONTINUE
175   CONTINUE
200   CONTINUE
      WRITE(6,230) P
230   FORMAT(/'  IRREDUCIBLE POLYNOMIAL P=1'/' P=',30I2/)
C
C
C     THIS PART CONSTRUCTS THE ARRAY MODULO.  ROW N OF THE ARRAY GIVES
C     THE POLYNOMIAL (OF DEGREE LESS THAN A) THAT THE POLYNOMIAL X**N IS
C     EQUIVALENT TO MODULO THE PRIME POLYNOMIAL P(X)
C
C
      DO 400 I=1,A
      MODULO(A,I)=P(I)
400   CONTINUE
      LIM1=2**A-2
      DO 500 N=APLUS1,LIM1
      NUM=2**(N-A)+1
      LIM2=2**(N-A+1)-1
450   NUM=NUM+1
      IF (NUM.GT.LIM2) STOP
      CALL CREATE(Q,NUM)
      CALL MULT(Q,P,R)
      R(N+1)=0
      DO 500 J=APLUS1,N
      IF (R(J).NE.0) GO TO 450
500   CONTINUE
      WRITE(6,550) N,Q,R
550   FORMAT(' EQUIVALENT POLYNOMIALS : X**N=Q*P+R  :  N= ',I5/
     C    '  Q= ',30I2/' R=',30I2/)
      DO 575 I=1,A
      MODULO(N,I)=R(I)
575   CONTINUE
600   CONTINUE
      WRITE(6,605) ((I,(MODULO(II,I2),I2=1,15),II=1,30)
605   FORMAT(///'  X**N MODULO THE PRIME POLYNOMIAL'//30(' N=',I4,5X,
     C   15I2/)////)
C     THIS PART TESTS A NUMBER OF POLYNOMIALS TO SEE IF THEY COULD
C     GENERATE THE FIELD.
C     A GENERATOR IS FOUND, AND THE VECTOR IS CREATED THAT CAN BE USED
C     TO CONSTRUCT THE ORTHOGONAL MATRIX
C
C
      NUM=2
620   NUM=NUM+1
      IF (NUM.GT.A2) STOP
      CALL CREATE(T,NUM)
      DO 640 I=1,A

      Q(I)=T(I)
640   CONTINUE
      N=1
      COL(1)=Q(1)
650   K=K+1
      IF (K.GT.A2) STOP
      CALL MULT(Q,T,S)
      CALL MODPOL(S,R)
      DO 680 I=1,A
      Q(I)=R(I)
680   CONTINUE
      COL(K)=Q(1)
      DO 690 I=2,A
      IF (Q(I).NE.0) GO TO 650
690   CONTINUE
      IF (K.NE.A2) GO TO 620
      WRITE(6,695) T
695   FORMAT(//'  GENERATOR T=1/2X,30I2//)
C
C
C     THIS PART OF THE PROGRAM TESTS THE MATRIX TO SEE IF THE
C     COLUMNS ARE ORTHOGONAL.
C
C
      WRITE(6,720) (COL(I),I=1,A2)
720   FORMAT(' COLUMN 1=  ',16011//(12X,16011//))
      DO 740 I=1,A2
      IF (COL(I).EQ.0) COL(I)=-1
740   CONTINUE
      DO 800 J=2,A2
      SUM=0
      DO 780 K=1,A2
      N=K+J-1
      IF (N.GT.A2) N=MOD(N,A2+1)+1
      SUM=SUM+COL(K)*COL(N)
780   CONTINUE
      IF (SUM.NE.-1) WRITE(6,790)
790   FORMAT(///' **** THE MATRIX IS NOT ORTHOGONAL ****')
      IF (SUM.NE.-1) STOP
800   CONTINUE
      WRITE(6,820)
820   FORMAT(////' **** THE MATRIX IS ORTHOGONAL ****')
      STOP
C
C
C
C     THIS SUBROUTINE MULTIPLIES TWO POLYNOMIALS OF DEGREE LESS THAN OR
C     EQUAL TO (A-1).  THE POLYNOMIALS W & X ARE MULTIPLIED, AND
C     THE RESULT IS Z.
C
C

      SUBROUTINE MULT(W,X,Z)
      INTEGER  W(30),X(30),Z(30)
      DO 1100 I=1,30
      Z(I)=0
1100  CONTINUE
      DO 1200 I=1,APLUS1

      DO 1200  J=1,APLUS1
      IF (W(I).EQ.1.AND.X(J).EQ.1) Z(I+J-1)=Z(I+J-1)+1
1200  CONTINUE
      LIMZ=2*A
      DO 1250 I=1,LIMZ
      Z(I)=MOD(Z(I),2)
1250  CONTINUE
      RETURN
C
C
C
C     THE ARRAY MODULO IS USED TO TAKE A POLYNOMIAL MODULO THE
C     PRIME POLYNOMIAL P.  WE FIND THE POLYNOMIAL Z (OF DEGREE LESS
C     THAN A) SUCH THAT Z = W(MODULO P).
C
C
      SUBROUTINE MODPOL(W,Z)
      INTEGER  W(30),Z(30)
      DO 1300 I=1,A
      Z(I)=W(I)
1300  CONTINUE
      LIMZ=2*A-2
      DO 1400 N=A,LIMZ
      IF (W(N+1).EQ.0) GO TO 1400
      DO 1350 J=1,A
      Z(J)=Z(J)+MODULO(N,J)
1350  CONTINUE
1400  CONTINUE
      DO 1430 J=1,A
      Z(J)=MOD(Z(J),2)
1430  CONTINUE
      RETURN
C
C
C
C     THIS SUBROUTINE CHECKS TO SEE IF THE POLYNOMIAL Z HAS A SOLUTION
C
C
      SUBROUTINE  SOLVE(Z)
      INTEGER  Z(30)
      SUM=0
      DO 1500 I=1,APLUS1
      SUM=SUM+Z(I)
1500  CONTINUE
      SUM=MOD(SUM,2)
      IF (SUM.EQ.0) SOLN='YES'
      IF (SUM.NE.0) SOLN='XXXXXX'
      IF (Z(1).EQ.0) SOLN='YES'
      RETURN

C     THIS SUBROUTINE IS USED TO CREATE A POLYNOMIAL Z CORRESPONDING TO THE
C     INTEGER N.  THE POLYNOMIAL IS OF ORDER LESS THAN OR EQUAL TO A+1.
C
C     NOTE THAT THIS IS WHERE THE UNIVAC FORTRAN FIELD (FLD) FUNCTION
C     IS USED TO EXTRACT THE BINARY BITS FROM THE NUMBER N AND CREATE
C     A POLYNOMIAL Z.  FOR EXAMPLE, IF N=11, THEN THIS IS EQUAL TO 1101
C     IN BINARY, AND THIS GIVES RISE TO THE POLYNOMIAL X**3+X**2+1.
C

      SUBROUTINE CREATE(Z,N)
      INTEGER  Z(30)
      DO 1600 I=1,APLUS1
      Z(I)=FLD(36-I,1,N)
1600  CONTINUE
      RETURN
C
C
C
      END
```

References

[1] Gurney, Margaret, "The Variance of the Replication Method for Estimating Variances for Grouped Strata," Technical Notes No. 3, U.S. Bureau of the Census, Washington, D.C., 1970, 7-12.

[2] Gurney, Margaret and Jewett, Robert, "Constructing Orthogonal Replications for Variance Estimation", Journal of the American Statistical Association (December, 1975).

[3] Herstein, I.N., Topics in Algebra, New York: Blaisdell (A Division of Ginn and Company), 1964.

[4] Keyfitz, Nathan (1957) "Estimation of Sampling Variance When Two Units are Selected From Each Strata", Journal of the American Statistical Association, 52, 503-510.

[5] McCarthy, P.J., "Replication: An Approach to the Analysis of Data from Complex Surveys," National Center for Health Statistics, Vital and Health Statistics, Public Health Service Publication No. 1000, Ser. 2, No. 14, Public Health Service, Washington, D.C.: U.S. Government Printing Office, 1966.

[6] Plackett, R. L., "Cyclic Intrablock Subgroups and Allied Designs," Sankhya, 8, No. 3 (1949) 275-6.

[7] Woodruff, Ralph, "A Simple Method for Approximating the Variance of a Complicated Estimate," Journal of the American Statistical Association, 66, June 1971.

CALCULATION OF CHI-SQUARE

Clifford J. Maloney

Food and Drug Administration

Pearson developed his chi-square test of significance in 1900 before it was pointed out to him by Soper that the then rediscovered Poisson distribution could be made the basis of a derivation. Fisher in 1922 showed that the multinomial could be viewed as a conditioned multiPoisson and made a "simplified" form of that formula the basis of an exact test. This paper shows that the "simplified" form of the test is a backward step and provides a computer program for the test.

I. Introduction: In 1900 Karl Pearson introduced the chi-square statistic to measure the adequacy of fit of a theoretical model to an observed frequency distribution. The technique quickly found other uses and continues to find widespread application in statistical practice; especially in the treatment of count data. The method however is subject to a number of deficiencies; one of the most troubling relates to applications where expected numbers are low. It is not fully realized that this disability is an artificial one, entirely a relic of its path of development.

The reader is referred elsewhere (Maloney 1978) for a detailed theoretical examination of the question. In this "how to" paper attention will be confined to the treatment of the R x C Two-way Contingency Table, including application to a particular example. In computer terms, the problem becomes one of performing a particular calculation, the probability of an observed outcome in terms of an assumed frequency distribution. The observed outcome is shown in Table I. The question to be answered is: are the data consistent with the assumption that all manufacturers of Pertussis vaccines are consistent with respect to the one property of vaccine potency? Lots released and lots not released are included in the table.

II. Pearson's Solution: Pearson showed that, if the expected cell count in every cell of the table (for example, Table I) is sufficiently large for a Poisson distribution to be adequately represented by a normal distribution with the same mean and variance, then the quantity

$$x^2 = \sum_m \frac{(x - m)^2}{m} \qquad (1)$$

calculated for all cells in the table could be referred to newly computed chi-square tables with the appropriate degrees of freedom. The effect of these tables is to avoid the repeated calculations required in the so called "exact" method of Fisher.

III. Fisher's Replacement: Perfecting a lead of Soner, earlier discussed by Pearson (1916), Fisher replaced the approximation (1) by the exact formula

$$p = \frac{\pi(x_{i.}!)\ \pi(x_{.j}!)}{N!\ \pi(x_{ij}!)} \qquad (2)$$

Formula (2) yields the probability of just one possible (or actual) outcome such as Table I. The calculation of (2) while more laborious then (1) is often still feasible, especially if tables of factorials and/or a computer is employed. The difficulty arises from the fact that many terms like (2) must be calculated. Only the one quantity in the denominator $\pi(X_{ij}!)$ changes at each step.

Still the volume of computation for a table as large as Table I (37 rows and 12 columns) quickly becomes expensive if not prohibitive. Even to count, not compute, the terms can be a formidable task (Gail and Mantel, 1977). Most of this effort results from terms of transcendently negligible probability, a condition not descernible in the form (2).

IV. Maloney's Revision: Fisher obtained his formula (2) as a "simplification" of

$$p = \frac{P(N,\ N)\ \pi P(x_{ij},\ m_{ij})}{\pi P(m_{i.},\ m_{i.})\ \pi P(m_{.j},\ m_{.j})} \qquad (3)$$

where each capital P stands for an individual term of the Poisson distribution

$$P(x,m) = \frac{m^x\ e^{-m}}{x!} \qquad (4)$$

Formula (2) is not however a simplification, except perhaps for extremely low total sample counts. This is due to two effects. First, in the form (3) rather than (2), the assistance of tables of the Poisson distribution (Molina 1947, General Electric 1962) can be exploited.

An even greater advantage of (3) over (2) is that each Poisson term P(x,m) can be individually calculated in (3) if the normal approximation is inadequate, or be replaced by the normal according to (1) if the approximation is adequate.

V. Detailed Procedure: For those cells where m is sufficently large, expression (1) is used. The requirement on m can be quite large. Every cell in Table I was in fact calculated by the procedure next to be described.

For each cell calculated by Poisson, the Poisson individual term probability for the observed outcome is calculated. Next, all terms whose probability is greater are calculated, subtracted from one and divided by two. Notice that this is a univariate calculation. For hand calculation cumulative Poisson tables would be used for this summation.

The result is the one tailed probability for the cumulated normal whose probability matches the Poisson accumulated probability for the possible outcome. Hence, we refer to the standard normal tables for the corresponding deviation in standardized units (or calculate it by computer). The square of that deviation is the contribution of that cell to the Pearson formula (1). The process continues until all cells are accounted for, when the sum in (1) is referred to standard chi-square tables (or again calculated by computer).

VI. Example Results: The unmodified Pearson formula gave a chi-square of 600.1 for Table I, highly significant. The method of this paper gave a value of 311.3, less than the 11 x 36 = 396 degrees of freedom.

Table II is a listing of the computer program. A printout of the detailed calculations is available.

VII. Acknowledgement: I am indebted for the data to Dr. Manclark, Food and Drug Administration; for the proprietary inverse normal routine to Mr. George Atta, National Institutes of Health, and for collaboration at every stage to Mrs. Lucille Carver, Food and Drug Administration.

VIII. References:

Fisher, R. A. (1922), "On the Interpretation of from Contingency Tables and the Calculation of P", Journal of the Royal Stat. Soc., A85, 87-94.

Gail, Mitchell and Mantel, Nathan (1977), "Counting the Number of r x c Contingency Tables with Fixed Margins", Journal of the Amer. Stat. Assn. (Dec.), 72, No. 360, 859-62.

General Electric Defense Systems Department (1962), "Tables of the Individual and Cumulative Terms of Poisson Distribution", New York: Van Nostrand.

Maloney, C. J. (1978), "Chi-Square for Small Expected Numbers", submitted to the Journal of the American Statistical Association.

Molina, E. C. (1947), "Poisson's Exponential Binomial Limit", New York: Van Nostrand.

Pearson, Karl (1916), "On the General Theory of Multiple Contingency with Special Reference to Partial Contingency", Biometrika, 11, 145-58.

Table I. Potency of Pertussis Vaccine Lots[a]

THD[b]	Manufacturer 3	4	5	6	7	8	9	10	11	12	13	14	Tot.
92	0	0	0	0	0	0	0	0	0	0	0	1	1
82	0	1	0	0	0	0	0	0	0	0	0	0	1
78	0	0	0	0	0	0	0	0	0	1	0	1	2
74	0	1	0	0	0	0	0	0	2	0	0	0	3
72	0	0	1	0	0	0	0	0	0	0	0	0	1
70	1	0	0	0	0	0	1	0	0	0	0	1	3
68	0	0	0	0	0	0	0	0	1	0	0	0	1
64	0	0	0	0	0	0	1	0	1	0	0	0	2
62	0	0	0	0	0	0	0	0	0	2	0	0	2
60	1	1	1	0	0	0	1	0	0	0	0	1	5
58	0	0	0	0	0	2	0	0	0	1	0	0	3
52	1	0	0	0	0	1	0	0	0	0	2	0	4
50	1	1	0	0	0	1	0	1	3	1	0	1	9
48	0	1	0	0	0	1	1	0	0	1	0	0	4
46	1	2	0	0	0	1	2	0	3	0	1	0	10
44	1	2	0	0	0	1	1	0	0	0	1	0	6
42	2	2	2	0	0	2	1	2	1	1	0	0	13
40	0	2	0	0	0	1	4	1	1	4	1	0	14
38	1	2	0	2	0	2	4	1	1	3	3	1	20
36	1	5	0	0	0	2	3	1	3	4	2	1	22
34	2	1	1	3	1	2	4	2	0	0	1	0	17
32	1	3	0	3	2	10	7	2	1	4	0	5	38
30	0	4	2	1	1	4	5	2	1	1	1	0	22
28	0	8	3	2	1	4	5	3	2	4	4	1	37
26	1	7	4	5	0	8	15	2	3	10	3	1	59
24	4	14	4	4	0	10	11	4	5	9	2	1	68
22	3	13	9	6	1	11	8	3	3	7	2	0	66
20	12	10	10	8	1	15	24	6	6	17	6	1	116
18	6	22	4	9	3	11	18	11	2	19	6	3	114
16	7	15	8	11	6	17	28	12	1	18	6	4	133
14	8	18	9	12	8	14	31	13	2	14	19	1	149
12	10	16	2	17	9	13	32	8	5	32	19	4	167
10	18	30	7	17	6	19	42	15	11	31	20	2	218
8	8	17	8	17	9	8	35	18	8	31	17	0	176
6	4	7	3	7	6	11	26	14	6	18	14	1	117
4	3	1	0	7	3	2	15	11	5	22	2	1	72
2	0	0	0	1	2	0	4	2	1	4	0	0	14
Tot.	97	206	78	132	59	173	329	134	78	259	132	32	709

a Tested from January 1957 to August 1965. Not necessarily released

b Total human dose

Table 2. Poisson Chi-Square Program

```
      DIMENSION T(80)
    5 FORMAT (3F8.5)
   10 FORMAT (1X,F10.8, I8)
   20 FORMAT (/4X,2HN=,I8,4X,3HIX=,I2,4X,2HM=,E15.8,4X,
     12HA=,E15.8,4X,2HB=,E15.8,2HP=,E15.8)
   30 FORMAT (//4X,5HT(J)=,E15.8,4X,7HT(J-1)=,E15.8/)
   34 FORMAT (//2HK=,I3,2HL=,I3)
   43 FORMAT (1X,2HK=,I8,2X,5HT(K)=,E15.8)
   50 FORMAT (///10X,14HPARAMETERS AT ,I8,13H'TH READING ',
     3I8,4X,F10.5,12HOUT OF RANGE//)
   55 FORMAT (2(4X,F10.5),34X,2(5X,F10.5))
   37 FORMAT (1H1,//1X,' OBSVD COUNT   ',' EXPCD COUNT',
     35X,'PROBABILITY',7X,'  DEVIATION   ',
     1'   CHI-SQUARE','   CHI-SQUARE SUM       '//)
      WRITE (15,37)
      REAL*8 M
      L=1000
      T(1) = 0.
      AL=50
      LEN = 40
      DO 64 N=1,L
   40 READ (1,5,END=999) D,X,M
      KNT = KNT + 1
      IF (KNT.LE.LEN) GO TO 606
      WRITE (15,37)
      KNT = 0
  606 CONTINUE
      IF (M.GT.AL) GO TO 45
      IX = X
      Z = DEXP(-M)
      IF (M.LE.0.) GO TO 25
      IF (X.LT.0.) GO TO 25
      IF (X.EQ.0.AND.M.LE.1.) GO TO 100
      IF (X.EQ.1.AND.M.LT.1.) GO TO 65
      IF (M.GE.X.AND.M.LT.X+1.) GO TO 35
      GO TO 15
   25 WRITE (15,50) N,IX,M
      GO TO 64
   35 CHI = 0.
      GO TO 111
   45 CHI = (X-M)*(X-M)/M
  111 SUM = SUM + CHI
      WRITE (15,55) X,M,CHI,SUM
      GO TO 64
   65 B = Z
      GO TO 75
  100 B = 0.
      GO TO 75
   15 T(1) = M
      DO 60 I=2,80
      T(I) =0.
   60 CONTINUE
      A = 0.
      IF (IX.GT.1) GO TO 88
      CK = 1.
      IF (IX.EQ.0) GO TO 85
      CK = M
      GO TO 85
   88 DO 66 J = 2,IX
      T(J) = T(J-1)*M/J
   66 CONTINUE
      IF (T(IX).LT.1.) GO TO 58
      A = 0.
      GO TO 11
   58 A = 1.
      GO TO 11
   11 CK = T(IX)
      IF (X.LE.1.) GO TO 85
      DO 80 K = 1,IX
      XK = K
      IF (T(K).LE.CK) GO TO 80
      A = A + T(K)
   80 CONTINUE
   85 I = IX + 1
      IF (X.EQ.0.) I=I+1
      DO 90 J = I, 60
      T(J) = T(J-1 )*M/J
      IF (T(J).LE.CK) GO TO 95
      A = A + T(J)
   90 CONTINUE
   95 B= A*Z
   75 P = 0.5*(1. - B)
      CALL MDNRIS(P,D,IER)
      IF (IER.NE.0) GO TO 42
      CHI = D*D
  222 SUM = SUM + CHI
      WRITE (15,200)  X,M,P,D,CHI,SUM
      GO TO 64
   42 WRITE (15,210) P
      GO TO 64
  200 FORMAT (3(4X,F10.5),5X,3(5X,F10.5))
  210 FORMAT (F10.5,2X, 'P LIES OUTSIDE OF RANGE')
   64 CONTINUE
  999 CALL EXIT
      STOP
      END
```

THE ROLE OF PSEUDO-RANDOM NUMBER GENERATORS IN THE HOPE-FILLED VARIANCE-REDUCING EFFORTS

G. Arthur Mihram, Ph. D.
Post Office Box N° 234
Haverford, Pennsylvania 19041

ABSTRACT

Considerable speculation has arisen in the recent literature [SIMULATION: STATISTICAL FOUNDATIONS AND METHODOLOGY (1972); OPERATIONS RESEARCH 21: 988 (1973); and, SIMULATION 22: 45 (1974), e.g.] of simulation methodology regarding the applicability of certain established procedures for reducing the variance of Monte Carlo estimators [MONTE CARLO METHODS (1964)]. In particular, antithetic variates and stratified sampling have been highly recommended to simulationists, though the direct applicability of antithetic variates to simular experimentation has been shown to be logically ill-founded [PROCEEDINGS, 1973 SUMMER COMPUTER SIMULATION CONFERENCE 1: 91 (1973); and COMPSTAT 1976: 256 (1976)].

THE PROBLEM

The present paper reiterates the current logical incompatibility of these Monte Carlo techniques and simular experimentation by depicting the generalised technique of Hammersley and Mauldon [6] as a requirement for an *ad hoc* pseudo-random generator. The promise of the mixed congruential generator for the purpose of variance reduction by either antithetic variates or stratified sampling is revealed, but it is noted that any suggestion for its widespread use in simular experimentation is weakly based. The use of stratification or antithetic variates in simular experimentation requires considerable knowledge of any stochastic model's behaviour as a transformation of the model's seed. Indeed, the use of either variance-reducing approach in simulation methodology requires an intimate knowledge of the model's transformation of the seed, for each environmental condition (input data save the seed) to be specified by its user.

The response of a dynamic and stochastic simulation is a random variable:

$$R(\vec{S};\ \vec{x},t) = r(\vec{x},t) + e(\vec{S};\ \vec{x},t), \qquad (1)$$

where

$$r(\vec{x},t) = E[R(\vec{S};\ \vec{x},t)], \qquad (2)$$

representing the simular response function (the mean value of the histogram, $g_R(y;\ \vec{x},t)$, which could be formed by collecting the model's responses from encounters of simular duration, t, and specified by a fixed vector, $\vec{x}$, of environmental conditions, yet over the entire set of admissible values of the model's juxtaposed random number seed, $\vec{S}$). [7]

Thus, for a fixed t and $\vec{x}$, the simular response, $R(\vec{S};\ \vec{x},t)$, is a transformation of the seed, $\vec{S}$; i.e., the whole model is a function. The histogram of these simular responses, $R(\vec{S};\ \vec{x},t)$, possesses not only a mean value, $r(\vec{x},t)$, but also a variance, σ^2. Thus, if one were to adhere to the Principium of Seeding [1] to define two separate encounters with the model and to provide therefore two statistically independent responses, $R_i \equiv R(\vec{S}_i;\ \vec{x},t)$, i = 1 and 2, then the simular response function can be estimated unbiassedly by

$$r^*(\vec{x},t) \equiv [R(\vec{S}_1;\ \vec{x},t) + R(\vec{S}_2;\ \vec{x},t)] / 2 \qquad (3)$$

an estimator having variance $\sigma^2 / 2$.

For the Monte Carlo evaluation of the finite Riemann integral,

$$I = \int_0^1 f(x)dx, \qquad (4)$$

Hammersley and Morton [8] devised a "simple" antithetic technique. Instead of computing the estimate,

$$I^* = [f(U_1) + f(U_2)] / 2 \qquad (5)$$

where U_1 and U_2 are independently and uniformly distributed random variables, Hammersley and Morton [8] recommended that, if f(x) be a continuous and monotone transformation of its argument, then, instead, one use

$$\bar{I} = [f(U_1) + f(1-U_1)] / 2. \qquad (6)$$

With

$$\mathrm{Var}[f(U)] = \mathrm{Var}[f(1-U)] \equiv \sigma^2,$$

then

$$\mathrm{Var}(I^*) = \sigma^2 / 2,$$

yet

$$\mathrm{Var}(I) = \sigma^2(1+\rho)/2,$$

where

$$\rho = \mathrm{Corr}[f(U), f(1-U)] < 0,$$

provided that $f(x)$ is a continuous and monotone function of its argument over the range: $a \leq x \leq b$.

Hence by taking advantage of particular information (continuity, monotonicity) regarding an integrand, one may use the antithetic variate, $1-U_1$, as the argument in a second evaluation of $f(x)$, then form their arithmetic mean, $\bar{I}$, to provide an unbiassed estimate of I, yet with a smaller variance than that of I^*.

In the event that the integrand, $f(x)$, is not monotone over the interval of integration, Hammersley and Mauldon [6] suggested a "generalised" antithetic technique [5]. Under this approach, one seeks a one-to-one transformation, $v(u)$, of the unit interval onto itself such that:

(a) except for at most a finite number of points, $dv/du = +1$; and,

(b) for $V = v(U)$, $f(U)$ and $f(V)$ shall not be positively correlated.

One would then compute,

$$\hat{I} = [f(U) + f(V)] / 2, \qquad (7)$$

with $V = v(U)$ selected so that V is itself a uniformly distributed variate, yet defined in an ad hoc fashion (i.e., specifically designed for the particular integrand, $f(x)$, at hand) so that

$$\mathrm{Corr}[f(U), f(v(U))] \leq 0.$$

By such an ad hoc procedure, one could assure that

$$\mathrm{Var}(\hat{I}) \leq \mathrm{Var}(I^*).$$

The recent literature [1, 2, 5] has questioned the widespread applicability of either the simple or the generalised antithetic approaches to efforts to reduce the variance of estimators of the simular response function. For a given model, can one define a generalised antithetic variate, $V = v(\vec{S})$, to be employed as the seed in a second encounter with a stochastic simulation model so that

$$\hat{r}(\vec{x},t) = [R(\vec{S}; \vec{x},t) + R(v(\vec{S}); \vec{x},t)] / 2 \qquad (8)$$

is an unbiassed estimator for the simular response function, an estimator with variance less than for $r^*(\vec{x},t)$ of equation (3) ?

MIXED CONGRUENTIAL GENERATORS

The mixed congruential generator [1, e.g.] provides exemplary functions of the generalised antithetic type. For a binary computer of standard computational word of b bits, the algorithm generates, from an integer, K, the pseudo-random integer, L, according to:

$$L \equiv (aK + c) \pmod{m}, \qquad (9)$$

where $m = 2^b$, $a \equiv 1 \pmod 4$ and c is chosen so that c and m have no common integral divisors greater then 1. Hence, if K is chosen randomly and uniformly from among the $m = 2^b$ integers between 0 and $2^b - 1$, inclusively, then the random variate, L, is also uniformly distributed, since, if employed sequentially, any properly defined mixed congruential generator issues exactly once every integer between 0 and $2^b - 1$, inclusively, before any among them is repeated.

In this sense, the mixed congruential generator acts as a shuffler of a deck of $2^b = m$ sequentially numbered cards. Two exemplary shufflings, each for the case $m = 8$, are depicted in Figures 1 and 2. One may note that the first generator permutes a deck of the integers, 0 through 7, yielding the random sequence: (7, 0, 1, 2, 3, 4, 5, 6); whereas the generator depicted in Figure 2 provides the permutation: (5, 2, 7, 4, 1, 6, 3, 0).

$L_1 \equiv (1K + 7) \pmod 8$

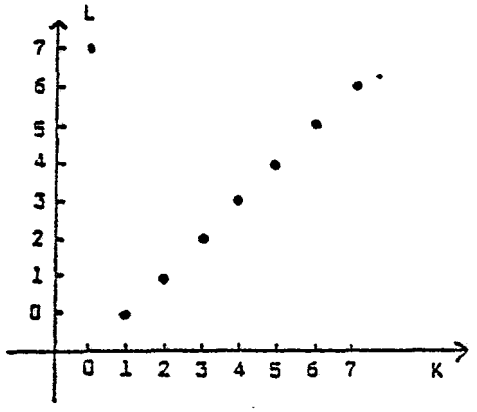


Figure 1. Mixed Congruential Generator ($a = 1$, $c = 7$)

$L_2 \equiv (5K + 5) \pmod 8$

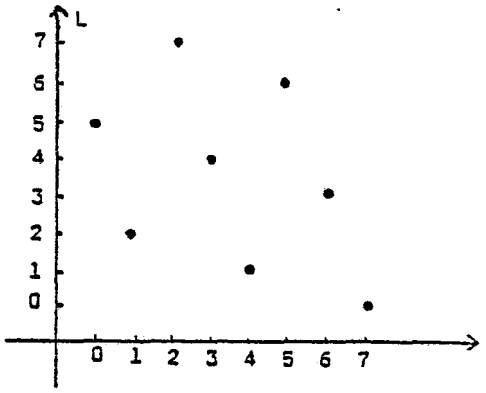


Figure 2. Mixed Congruential Generator ($a = 5$, $c = 5$)

One may view the mixed congruential generator as a transformation, $V = v(U)$, of a continuously uniform random variable, U:

$$V = v(U) \equiv [L + (mU - K)] / m, \qquad (10)$$

wherein $K = [\,[mU]\,]$, the greatest integer less than or equal to mU, and L is obtained from K via the mixed congruential generator of Equation (9). Two such continuous, generalised antithetic variates are depicted in Figures 3 and 4, corresponding to the mixed congruential generators depicted in Figures 1 and 2,

respectively.

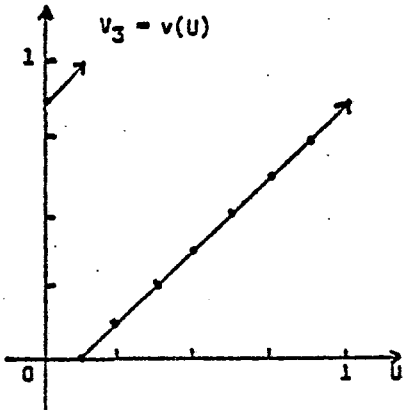


Figure 3. Generalised Antithetic Variate (a = 1, c = 7)

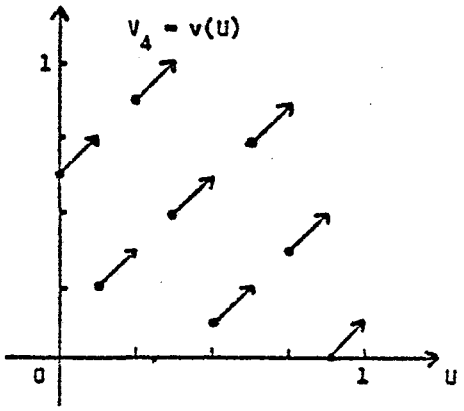


Figure 4. Generalised Antithetic Variate (a = 5, c = 5)

One notes that the functions depicted indeed satisfy the two conditions, of Hammersley and Mauldon [6], for V = v(U) to be a generalised antithetic variate. (Cf: Conditions preceeding Equation (7).) One may note that the discrete random variates, L_1, and L_2, of Figures 1 and 2, respectively, have the property of being discretely uniform variates, with

$$E(L_1) = E(L_2) = E(K) = m^{-1} \cdot \sum_{i=0}^{m-1} i = (m-1)/2 = 7/2$$

and

$$Var(L_1) = Var(L_2) = Var(K) = 21/4,$$

while

$$Cov(K, L_1) = +7/4 \ (> 0),$$

yet

$$Cov(K, L_2) = -9/4 \ (< 0).$$

Similarly, the corresponding continuous random variates, V_3 and V_4, of Figures 3 and 4, respectively, are continuously uniformly distributed, with the properties that

$$E(V_3) = E(V_4) = E(U) = 1/2,$$

and

$$Var(V_3) = Var(V_4) = Var(U) = 1/12,$$

while

$$Cov(U,V_3) = \int_0^{1/8} u(u+7/8)du + \int_{1/8}^{1} u(u-1/8)du - 1/4$$

$$= 11/384 = 0.0287 \ (> 0),$$

yet

$$Cov(U,V_4) = -(104/3072) = -0.034 \ (< 0).$$

In terms of correlation coefficients, one may list the results:

$$Corr(K, L_1) = +1/3,$$

$$Corr(K, L_2) = -3/7,$$

$$Corr(U, V_3) = +11/32, \text{ and}$$

$$Corr(U, V_4) = -0.406.$$

the first two of which represent the autocorrelation of the two respective mixed congruential generators (autocorrelation of lag 1).

VARIANCE REDUCTION

Thus the mixed congruential generators can be employed to transform an uniformly distributed variate into a second uniformly distributed variate, though the induced correlation between the two variates may be positive or negative. The generator depicted in Figures 1 and 3 constitutes a "stratified sampling" [10] approach to variance reduction, whereas the generator provided in Figures 2 and 4 is a more exemplary generalised antithetic variate.

One may note that the choice of a generator, which will induce a negative correlation between its pair of uniformly distributed variates, depends upon one's selection of the generator's additive and multiplicative constants (c and a, respectively). Yet, in Monte Carlo investigations, this is precisely what Hammersley and Mauldon [6] proposed; viz., that, for a given integrand, f(x), of integral, I, of Equation (4), one shall need to define an ad hoc generalised antithetic variate. For the binary computer of word size of b bits, there are 2^{b-1} choices for the additive constant (c) and then 2^{b-2} choices for the multiplicative constant (a), so that, conceptually, one could search through the $2^{2b-3} = m^2/8$ mixed congruential generators so as to locate that one pair of constants (a, c) for which the correlation is minimised.

One should note, however, that one must proceed in a somewhat ad hoc fashion, determining whether

$$Corr[f(U), f(V)] < 0,$$

for V as defined in Equation (10). Without an algorithm for conducting this search expediently, one could rightfully question the practical utility of antithetic variates in Monte Carlo analyses.

One may note that the simular response function can be construed as an integral:

$$r(x,t) = \int_0^1 R(u; \vec{x}, t)du,$$

where $U = \vec{S}/m$ is an uniformly distributed variate

whenever the stochastic model's juxteposed seed, $\vec{S}$, is selected in accordance with the Principium of Seeding [1]. In this sense, experimentation with a particular, dynamic, and stochastic simulation model is indeed analogous to the Monte Carlo evaluation of an integral. Yet, for a given model, the unknown integral, $I = r(\vec{x},t)$, depends not only upon the model (which is the "integrand") but also upon the particular set of "parameters" employed [i.e., upon the input data specifications for the model's environmental conditions ($\vec{x}$) and simular duration (t)].

CONCLUSION: (VARIANCE REDUCTION OR VARIANCE REPRODUCTION?)

Without an algorithm delineating the procedure by which one selects a generalised antithetic variate for seeding the second of two encounters with a dynamic and stochastic simulation model, one must question any widespread or "blind" application of antithetic variates in simular experimentation. Indeed, if antithetic variates are to prove useful in experiments with stochastic, computerised models, then it would appear that one must employ, in successive pairs of model encounters, a pair of antithetic seeds, one pair for each specification (x,t) of the model's input data.

Perhaps, as scientists employing simulation to model reliably the systemic problems which currently challenge us, we should focus attention on an important aspect of simular credibility: model confirmation. Since a dynamic and stochastic simulation model is designed to mime the behaviour of its simuland, then the statistical properties of the simular responses must be compatible with the corresponding measures made on the modelled system.

Typically, model confirmation is conducted via the statistician's collection of two-sample tests [9]. In particular, homogeneity of the variance of the simular responses and that of the actual system's corresponding yields is of paramount import. Thus, variance reproduction is a topic of considerably greater import to the systemic scientist than is variance reduction.

Model confirmation must be attended if indeed simulation is to earn its designated title: The Methodology of the Scientist of Systems.

REFERENCES

1. Mihram, G. Arthur, SIMULATION: STATISTICAL FOUNDATIONS AND METHODOLOGY, New York: Academic Press, 1972.

2. Mitchell, Bill, "Variance Reduction by Antithetic Variates in G/G/1 Queuing Simulations," OPERATIONS RESEARCH, 21, 988-997, 1973.

3. Kleijnen, Jack P. C., "A Note on Sampling Two Correlated Variables," SIMULATION, 22, 45-46, 1974.

4. Hammersley, J. M. and D. C. Handscomb, MONTE CARLO METHODS, New York: John Wiley, 1964.

5. Mihram, G. Arthur, "On Antithetic Variates," PROCEEDINGS, 1973 SUMMER COMPUTER SIMULATION CONFERENCE, 1, 91-95, 1973.

6. Hammersley, J. M. and J. G. Mauldon, "General Principles of Antithetic Variates," PROCEEDINGS, CAMBRIDGE PHILOSOPHICAL SOCIETY, 52, 476-481, 1956.

7. Mihram, G. Arthur, "A Glossary of Simulation Terminology," JOURNAL OF STATISTICAL COMPUTATION AND SIMULATION, 1, 35-44, 1971.

8. Hammersley, J. M. and K. W. Morton, "A New Monte Carlo Technique: Antithetic Variates," PROCEEDINGS, CAMBRIDGE PHILOSOPHICAL SOCIETY, 52, 449-475, 1956.

9. Mihram, G. Arthur, "Simular Credibility," PROCEEDINGS, 1973 SUMMER COMPUTER SIMULATION CONFERENCE, 1, 96-99, 1973.

10. Gaver, D. P., "Statistical Methods for Improving Simulation Efficiency," PROCEEDINGS, THIRD CONFERENCE ON APPLICATIONS OF SIMULATION, 38-46, La Jolla, Calif.: Society for Computer Simulation, 1969.

*

* *

Software for Iteratively Reweighted Least Squares Computations

Stephen C. Peters
Virginia C. Klema
MIT Center for Computational Research in Economics and Management Science

Paul Holland
Educational Testing Service

ABSTRACT

Robust linear regression problems can be solved by the method of iteratively reweighted least squares. Stable numerical algorithms applicable to this problem are known for computing the ℓ_1 and ℓ_2 starting estimates, the subsequent iterations, and a scale free convergence criteria. A system of semi-portable Fortran subroutines embodying these techniques is described. Eight weight functions, numerical rank determination, various regression statistics, a stem-and-leaf display, and an interactive driver are included.

Development of this software and associated research was supported by the National Science Foundation under Grants DCR75-08802 and MCS76-11989.

0. Introduction

Much study and a great deal of progress has been made towards finding robust alternatives to the classical least squares fitting techniques. These robust procedures are not adversely affected by moderate departures from assumptions concerning the disturbances, nor by a small number of truly discrepent (perhaps misrecorded) observations. For the robust fitting of linear regressions, the method of iteratively reweighted least squares has proven quite successful. Stable and efficient algorithms for computing the solutions to weighted least squares problems are available. We have carefully assembled these algorithms in a system of subroutines which reliably solve iteratively reweighted least squares problems.

1. The Robust Linear Regression Problem

We propose to fit the linear model

$$Y = X\beta + \varepsilon \tag{1.1}$$

where Y is the n-vector of observations on the dependent variable, $Y = (y_1, y_2, \ldots, y_n)^T$,
X is the nxp matrix of carriers or independent variables, $X = (x_1, x_2, \ldots, x_n)^T$,
β is the p-vector of coefficients to be determined, and
ε is an n-vector of random errors or disturbances.

The fit is determined from the minimization

$$\min_{\beta} \sum_{i=1}^{n} \rho_c\left(\frac{y_i - x_i\beta}{s}\right) \tag{1.2}$$

where $\rho_c(\cdot)$ is a loss function satisfying $\rho_c(0) = 0$ and $\rho_c(u) \leq \rho_c(t)$ if $|u| < |t|$. s is a scale or size for the residuals $y_i - x_i\beta$. When $\rho_c(t) = t^2$, (1.2) is ordinary least squares regression, s may be taken as an arbitrary (non-zero) constant, and a variety of computational techniques are available to affect a stable solution (see for example [9] or [15]).

When ρ_c is a robust loss function, for example

$$\rho_c(t) = \begin{cases} t^2 & |t| \le c \\ c|t| - c^2 & |t| > c \end{cases}, \qquad (1.3)$$

(1.2) generally becomes a non-linear minimization problem. Because robust ρ_c are not usually scale invariant, selection of s requires special attention. The "tuning constant" or "robustness parameter", c, directly effects the departure of the solution from least squares and is chosen prior to the minimization with regard to efficiency.

Solution of (1.2) for the robust loss case may be obtained by choosing a starting guess, $\beta^{(0)}$, finding a scale s as a robust function of the residuals $y_i - x_i\beta^{(0)}$, and taking Gauss-Newton iterations:

$$\beta^{(k+1)} = \beta^{(k)} + s(X^T\rho_c''^{(k)}X)^{-1}X^T\rho_c'^{(k)} \qquad (1.4)$$

where $\rho_c'^{(k)} = \left[\rho_c'\left(\frac{y_1 - x_1\beta^{(k)}}{s}\right), \ldots, \rho_c'\left(\frac{y_n - x_n\beta^{(k)}}{s}\right)\right]^T$

(primes indicating derivatives)

and $\rho_c''^{(k)}$, similarly defined, is an n x n diagonal matrix. For non-convex ρ_c, the solution thus obtained may be only a local minimum of (1.2). It may also be desirable to recompute the residual scale as the iteration proceeds, but here we do not include s formally in the minimization.

To ease the computational expense of forming $\rho_c''^{(k)}$ and to avoid those situations where $\rho_c''^{(k)}$ possesses discontinuities, approximations to the term $(X^T\rho_c''^{(k)}X)$ which preserve positive semi-definiteness are often made. Huber [13] has suggested iterates of the form

$$\beta^{(k+1)} = \beta^{(k)} + s(X^TX)^{-1}X^T\rho_c'^{(k)} \qquad (1.5)$$

approximating $\rho_c''^{(k)}$ by the identity matrix. This method has the advantage that once the least squares problem implicit in (1.5) is first solved, i.e.,

$$s\rho_c'^{(0)} = X(\beta^{(1)} - \beta^{(0)}) \qquad (1.6)$$

subsequent iterates may be obtained at a fraction of the initial cost.

The method we use more closely approximates ρ_c'' as

$$\rho_c''(u) \approx \frac{\rho'(u)}{u} \qquad (1.7)$$

(this is just the slope of the secant line joining $\rho_c'(u)$ to the origin). If we define the weighting function

$$w_i\left(\frac{y_i - x_i\beta^{(k)}}{s}\right) \equiv \frac{\rho_c'\left(\frac{y_i - x_i\beta^{(k)}}{s}\right)}{\left(\frac{y_i - x_i\beta^{(k)}}{s}\right)} \qquad (1.8)$$

and the diagonal matrix $W^{(k)}$, to be formed from the w_i, then our approximation to (1.4) is

$$\beta^{(k+1)} = \beta^{(k)} + s(X^TW^{(k)}X)^{-1}X^TW^{(k)}\left(\frac{Y - X\beta^{(k)}}{s}\right) \qquad (1.9)$$

A little simplification yields

$$\beta^{(k+1)} = (X^TW^{(k)}X)^{-1}X^TW^{(k)}Y, \qquad (1.10)$$

Thus, at each iteration we solve the weighted least squares problem:

$$W^{\frac{1}{2}(k)}Y = W^{\frac{1}{2}(k)}X\beta^{(k+1)} \qquad (1.11)$$

where the non-negative weights are functions of the scaled residuals determined at the previous iteration. The method of weighted least squares is widely known, the application of iteratively reweighted least squares to the robust linear regression problem was suggested in Andrews [2] and Beaton and Tukey [4]. For further theoretical discussion see Holland and Welsch [12] and the references therein.

The software we have implemented solves the robust linear regression problem with (robustly determined) fixed scale by the method of iteratively reweighted least squares. The weight functions we provide as part of the software are listed in Table 1.

2. Starts, Steps, and Stopping

The iterative process requires a start, $\beta^{(0)}$, which can be obtained in our programs from the ordinary least squares solution of $Y = X\beta^{(0)}$, from the overdetermined solution in the ℓ_1 norm of $Y = X\beta^{(0)}$ (which corresponds to least absolute residuals regression), from previous iterations of iteratively reweighted least squares

Table 1

Weight functions (where u = scaled residual), range and default tuning constants.

Name	w(u)		Range	Tuning Constant*
ANDREWS	$w_A(u)$ =	$\sin(u/A)/(u/A)$	$\|u\| \le \pi A$	A = 1.339
		0	$\|u\| > \pi A$	
BIWEIGHT	$w_B(u)$ =	$[1 - (u/B)^2]^2$	$\|u\| \le B$	B = 4.685
		0	$\|u\| > B$	
CAUCHY	$w_C(u)$ =	$1/(1+ (u/C)^2)$		C = 2.385
FAIR	$w_F(u)$ =	$1/(1+ \|u/F\|)$		F = 1.400
HUBER	$w_H(u)$ =	1	$\|u\| \le H$	H = 1.345
		$H/\|u\|$	$\|u\| > H$	
LOGISTIC	$w_L(u)$ =	$(\tanh(u/L))/(u/L)$		L = 1.205
TALWAR	$w_T(u)$ =	1	$\|u\| \le T$	T = 2.795
		0	$\|u\| > T$	
WELSCH	$w_R(u)$ =	$e^{-(u/R)^2}$		R = 2.385

*These are default values for the tuning constants which are designed to have 95% asymptotic efficiency with respect to ordinary least squares when the disturbances come from the normal or Gaussian distribution and the scaling function converges to the standard deviation of that disturbance distribution.

(perhaps using a different weight function, scale, etc.), or from a user defined subroutine. The starting scale value is determined as the median of the absolute values of the non-zero residuals $Y-X\beta^{(0)}$. The modularity of the software makes readily possible the inclusion of additional residual scaling functions (e.g., the inter-quartile range of the residuals).

The least squares start, and iterations (steps) subsequent to any start, are computed by the numerically stable and computationally efficient method of orthogonal factorization. Such techniques are generally advisable for solving least squares problems and are especially recommended here where, at any iteration, re-weighting may down-weight rows to the extent that their effective deletion creates rank degeneracy. Specifically, we form an orthogonal factorization of X (or $W^{\frac{1}{2}(k)}X$) by Householder transformations with column pivoting:

$$QXP = \begin{bmatrix} R \\ 0 \end{bmatrix} \qquad (2.1)$$

where Q is an orthogonal matrix, P is a permutation matrix, and R is upper trapezoidal. Unless X is exactly singular, R is an upper triangular matrix whose conditioning with respect to least squares solution is identically that of X. The condition of R is then estimated in $O(p^2)$ operations by the algorithm of Cline et.al. [5]. If the condition of X as judged by this estimate is sufficiently small, we go ahead and solve the (weighted) least squares problem. Otherwise we form the computationally more expensive singular value decomposition of X, determine the rank of X with respect to the certainty of the data [10], and then solve the (weighted) least squares problem.

We encourage the user to supply an indication of the certainty of the data when X is input, and to examine those singular values which are produced in the course of iteration to satisfy himself with the strength of the rank. We do, however, provide a conservative default determination of rank associated with the condition of the matrix at each iteration relative to the square-root of the precision of the computing machine being used. Explicitly, when the condition estimate of R exceeds $\varepsilon^{-\frac{1}{2}}$ (where ε is the relative precision of the machine arithmetic) the singular value decomposition is invoked and the rank of the matrix is taken to be

k determined so that

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_k \geq \epsilon^{\frac{1}{2}} > \sigma_{k+1} \geq \ldots \geq \sigma_p \geq 0 \qquad (2.2)$$

where the σ_i are the singular values of X.

To test for convergence of the iterative process we use the covergence criteria suggested by Dennis [6]. At convergence the weighted residuals, $W^{\frac{1}{2}(k)} r^{(k)} = W^{\frac{1}{2}(k)}(Y - X\beta^{(k)})$, should be orthogonal to the reweighted matrix of carriers $W^{\frac{1}{2}(k)} X$. As a scale free measure of orthogonality we examine the cosine of the angle between $W^{\frac{1}{2}(k)} r^{(k)}$ and the j^{th} column of $W^{\frac{1}{2}(k)} X$, for j=1,...,p, which is

$$(W^{\frac{1}{2}(k)} X_j)^T (W^{\frac{1}{2}(k)} r^{(k)}) \Big/ \| W^{\frac{1}{2}(k)} X_j \|_2 \, \| W^{\frac{1}{2}(k)} r^{(k)} \|_2 \qquad (2.3)$$

where $\|\cdot\|_2$ is the Euclidean norm. This is a scale free measure of the gradient for the minimization.

3. Software Particulars

The software we provide to compute the iteratively reweighted least squares solutions consists of 17,000 lines of Fortran code, comments, and documentation for use. The software is modular, containing over 80 subroutines constructed to the portability standard described in [14], and has been checked by the PFORT Verifier [16]. Included are the control statements (Fortran comments) required by the IMSL Fortran Converter [1] to produce source versions for IBM, CDC, Burroughs, Honeywell, DEC PDP-10, and Univac machines.

To compute the least squares start and iterations the subroutine collection uses orthogonal factorizations by Householder transformations, or the singular value decomposition from EISPACK II [8]. CL1 [3] computes the ℓ_1 start. The weight functions have been carefully coded to be resistant to underflow and its side-effects. We supply an interactive driver with online "help" facilities for use in the time-sharing environment although the subroutines are just as useful in batch systems. In addition to the robust least squares results (coefficient estimates, residuals, and final weights), many other statistics are (optionally) reported. These include (weighted) sum of squared residuals, (weighted) standard error of the regression, (weighted) R-squared and F statistics, and sum of absolute residuals. The diagonal elements of the least squares projection matrix ("hat" matrix) are also (optionally) computed. Stem-and-leaf displays [11] of the residuals, weights, and hat matrix diagonals can be produced. A group of test problems selected from [7] are included.

The software is available through IMSL Inc. at the following address:

International Mathematical and Statistical Libraries, Inc.
Sixth Floor GNB Building
7500 Bellaire
Houston, Texas 77036 USA

4. Acknowledgements

We are grateful for the assistance of Roy Welsch in providing statistical perspectives of robust regression. Neil Kaden and David Coleman programmed and tested most of the subroutines. The National Science Foundation provided funding.

References

[1] Aird, T.J., "The Fortran Converter User's Guide," IMSL, 1975.

[2] Andrews, D.F., "A Robust Method for Multiple Linear Regression," Technometrics 16 (1974).

[3] Bartels, R., and Conn, A., "Linearly Constrained Discrete ℓ_1 Problems," Johns Hopkins University, Technical Report #248, June 1976.

[4] Beaton, A.E., and Tukey, J.W., "The Fitting of Power Series, Meaning Polynomials, Illustrated on Band-Spectroscopic Data", Technometrics 16 (1974).

[5] Cline, A., Moler, C., Stewart, G.W., and Wilkinson, J.H., "On an Estimate for the Condition Number of a Matrix," informal manuscript, 1977.

[6] Dennis, J.E., "Non-Linear Least Squares and Equations," in The State of the Art of Numerical Analysis, D. Jacobs, editor, Academic Press, 1977.

[7] Draper, N.R., and Smith, H., Applied Regression Analysis, John Wiley and Sons, Inc., 1966.

[8] Garbow, B.S., Boyle, J.M., Dongarra, J.J., and Moler, C.B., Matrix Eigensystem Routines - Eispack Guide Extension, Springer-Verlag, Lecture Notes in Computer Science, 51, 1977.

[9] Golub, G.H., "Matrix Decompositions and Statistical Calculations," in Statistical Computation, R.C. Milton and J.A. Nelder, editors, Academic Press, 1969.

[10] Golub, G.H., Klema, V., and Stewart, G.W., "Rank Degeneracy and Least Squares Problems," University of Maryland TR-456, 1976, Stanford University, STAN-CS-76-559, 1976, National Bureau of Economic Research, Inc., Working Paper 165, 1977.

[11] Hoaglin, D.C., and Wasserman, S., "Automating Stem-and-Leaf Displays," National Bureau of Economic Research, Inc., Working Paper 109, 1975.

[12] Holland, P., and Welsch, R., "Robust Regression Using Iteratively Reweighted Least Squares," Communications in Statistics: Theory and Methods, 1977.

[13] Huber, P.J., "Robust Statistical Procedures," Regional Conference Series in Applied Mathematics, Society for Industrial and Applied Mathematics, 1977.

[14] Kaden, N., and Klema, V., "Guidelines for Writing Semi-Portable Fortran," National Bureau of Economic Research, Inc., Working Paper 130, 1976.

[15] Lawson, C.L., and Hanson, R.J., Solving Least Squares Problems, Prentice-Hall, Inc. 1974.

[16] Ryder, B.G., "The Fortran Verifier: User's Guide," Computing Science Tech. Report 12, Bell Telephone Labs., 1975.

SURVEY SOFTWARE -- A METHOD FOR HANDLING THE VARIANCE PROBLEM

Clark Readler

Systems Software Division, U.S. Bureau of the Census
Suitland, Maryland

Abstract

Sample survey designs are generally complex, and require complicated variance calculations. Most survey software packages either ignore the problem, or provide an inadequate capability to obtain valid estimates of precision (standard errors). A general method is presented for handling this requirement.

INTRODUCTION

Most probability sample surveys are not based on simple random or unrestricted random samples. Rather they are usually stratified, often clustered, and frequently employ ratio or regression estimates. Some surveys employ a "probability proportional to size" sampling scheme. There is also a growing trend toward post stratification ratio adjustments. Using various combinations of these designs produces extremely complicated variance formulas.

Unless a "most correct" estimation procedure is used, it is infeasible to envision a routine that calculates the variances for more than a few variance formulas. It is also not feasible to envision a routine utilizing the "most correct" approach that would handle every survey design. [1]

CALCULATING VARIANCES

The problem of variance estimation may be answered by extending the language of the tabulating systems. This extension will be presented in the context of the GTS language.

The GTS language allows easy construction of tables for any survey design. It is a simple matter to construct tables containing sums of observations, sums of squares, sums of cross products, and counts of the number of observations. The extension of the language will include an APL like ability to perform table, column, and cell computations, (i.e. table compute, and column compute statements). The commands will allow tables to be conceptualized as "unitary" variables and plugged into any variance equation.

CURRENT TABULATING SYSTEMS

Numerous tabulating systems have been developed. Among them are GTS (Generalized Tabulating System), developed at the U.S. Bureau of the Census, TPL (Table Producing Language) developed at the Bureau of Labor Statistics, and CENTS AID II (Census Tabulation System Aid) developed by Data Use and Access Laboratories. These software systems differ from statistical packages, such as SPSS, in that they are actual tabulating languages and not a series of statistical procedures.

The tabulating systems are designed to create production tables and tables for analyst review. These languages can produce tables of first order estimates for almost any survey design (i.e. means, population totals, percents, etc.). However none of these systems provides an easy method for variance calculation, except for the most simple designs.

EXTENSIONS TO A TABULATING LANGUAGE

Let R be the real numbers and $f:R^k \rightarrow R$ where k=1 or 2. Suppose that S_{mxn}, T_{pxq}, U_{rxs} are tables whose dimensions are indicated by their subscripts.

Then for k = 1

$S_{mxn} = f(T_{pxq})$	<=>	
$s(i,j) = f(t(i,j))$	when	m=p, n=q
$s(i,j) = f^{(n)}(t(i,1))$	when	m=p, q=1, and any n
$s(i,j) = f^{(m,n)}(t(1,1))$	when	p=1, q=1, and any m,n

For k = 2

$S_{mxn} = f(T_{pxq}, U_{rxs})$

$s(i,j) = f(t(i,j),u(i,j))$ when m=p=r, n=q=s

$s(i,j) = f(t(i,j),u(i,1))$ when m=p=r, n=q, s=1

$s(i,j) = f(t(i,1),u(i,j))$ when m=p=r, n=s, q=1

$s(i,j) = f^{(n)}(t(i,1),u(i,1))$ when m=p=r, q=1, s=1 and any n

$s(i,j) = f(t(i,j),u(1,1))$ when m=p, n=q, r=s=1

$s(i,j) = f^{(n)}(t(i,1),u(1,1))$ when m=p, q=r=1, s=1 and any n

$s(i,j) = f(t(1,1),u(i,j))$ when m=r, n=s, p=q=1

$s(i,j) = f^{(n)}(t(1,1),u(i,1))$ when m=r, p=q=1, s=1 and any n

$s(i,j) = f^{(m,n)}(t(1,1),u(1,1))$ when p=q=r=s=1 and any m,n

where s(i,j), t(i,j) and u(i,j) are elements in the i^{th} row and the j^{th} column of tables S, T, U respectively.

Example:

Let $T1 = \begin{bmatrix} a_1 & b_1 \\ c_1 & d_1 \end{bmatrix}$ $T2 = \begin{bmatrix} a_2 & b_2 \\ c_2 & d_2 \end{bmatrix}$

Then in the context of the GTS Language we have

Table Compute T = T1*T2 => $T = \begin{bmatrix} a_1*a_2 & b_1*b_2 \\ c_1*c_2 & d_1*d_2 \end{bmatrix}$

Table Compute T = T1/T2(1) => $T = \begin{bmatrix} a_1/a_2 & b_1/a_2 \\ c_1/c_2 & d_1/c_2 \end{bmatrix}$

Table Compute T = T1+T2(1,2) => $T = \begin{bmatrix} a_1+b_2 & b_1+b_2 \\ c_1+b_2 & d_1+b_2 \end{bmatrix}$

Table Compute T = 7 => $T = \begin{bmatrix} 7 & 7 \\ 7 & 7 \end{bmatrix}$

EXAMPLE

To illustrate the power of these commands, consider a survey designed to estimate the length of financing for various export commodity groups. The exports are classified into five commodity groups. Each export is also classified into one of five sub-groups. A random sample is drawn independently in each commodity group.

SUB-GROUPS

STRATA	CASH	2 yr.	2-4 yr.	4-8 yr.	8 yr.
Commodity 1					
Commodity 2					
Commodity 3					
Commodity 4					
Commodity 5					

The design of the survey is a stratified random sample with sub-groups producing a five by five table of estimates.

The estimate of the total $\hat{X}_{ij}$ for each sub-group within strata is given by $\hat{X}_{ij} = \frac{N_i}{n_i}\Sigma y_{ijk}$ where y_{ijk} denotes the k^{th} observation in the y^{th} sub-group within the i^{th} commodity group.

N_j = population total for commodity group$_i$

n_i = sample total for commodity group$_i$

$$Var(\hat{X}_{ij}) = N_i^2/n_i*(1-n_i/N_i)*(\Sigma y_{ijk} - \frac{(\Sigma y_{ijk})^2}{n_i})/(n_i-1)$$

Let $T1 = \begin{bmatrix} N_1 & n_1 \\ N_2 & n_2 \\ N_3 & n_3 \\ N_4 & n_4 \\ N_5 & n_5 \end{bmatrix}$

$T2 = \begin{bmatrix} \Sigma y_{11k} & \Sigma y_{12k} & \cdot & & \Sigma y_{15k} \\ \Sigma y_{21k} & & & & \cdot \\ \cdot & & \Sigma y_{33k} & & \cdot \\ & & & & \\ \Sigma y_{51k} & & \cdot & & \Sigma y_{55k} \end{bmatrix}$

$T3 = \begin{bmatrix} \Sigma y_{11k}^2 & \Sigma y_{12k}^2 & & & \Sigma y_{15k}^2 \\ \Sigma y_{21k}^2 & \cdot & \cdot & & \cdot \\ & & \Sigma y_{33k}^2 & & \\ \cdot & \cdot & & \cdot & \\ \Sigma y_{51k}^2 & & \cdot & & \Sigma y_{55k}^2 \end{bmatrix}$

Using the table compute statement the tables of estimates $T(\hat{X}_{ij})$ and $T(\sigma_{ij})$ can be computed by

Table compute T3 = sqrt(T1(1)*T1(1)/T1(2)*(1-T1(2)/ T1(1))*(T3-(T2*T2/T1(2))/(T1(2)-1))

Table compute T2 = T1(1)/T1(2)*T2

T2 and T3 have been transformed to:

$T2 = \begin{bmatrix} \hat{X}_{11} & \hat{X}_{12} & \cdot & \cdot & \hat{X}_{13} \\ \hat{X}_{21} & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \hat{X}_{33} & & \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \hat{X}_{51} & \cdot & \cdot & & \hat{X}_{55} \end{bmatrix}$

$$T3 = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdot & \cdot & \sigma_{13} \\ \sigma_{21} & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \sigma_{33} & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \sigma_{51} & \cdot & \cdot & \cdot & \sigma_{55} \end{bmatrix}$$

EQUIVALENT FORTRAN STATEMENTS

If a custom FORTRAN program was written, the same concepts (creating table structures) would be employed. The final FORTRAN statements would be:

```
      DO  100  i = 1, 5
      DO  100  j = 1, 5

      T3(i,j) = sqrt(T1(i,1)*T1(i,1)/T1(i,2)*
                (1-T1(i,2)/T1(i,1))*(T3(i,j)-T2(i,j)*
                T2(i,j)/T1(i,2))/(T1(i,2)-1))

100   continue


      DO  200  i = 1, 5
      DO  200  j = 1, 5

      T2(i,j) = T1(i,1)/T1(i,2)*T2(i,j)

200   continue
```

The table compute command performs the same function as nested DO loops, but is conceptually easier to comprehend. Hence, the table compute, and analogous column, row, and cell compute commands allow a FORTRAN like ability for computing second order estimates.

DISADVANTAGES

Although the parsing and implementation algorithms are quite simple a lot of computer core will be utilized. If the target machine for implementation does not contain a paging mechanism, then about 80K of core would be required with a user limit of 60000 table cells.

CONCLUSION

There is no feasible way to create a routine that will compute the variances for all possible survey designs. The only possible answer to the problem of variance calculation is to allow a table manipulation or matrix computation ability. The table computation commands can be compared to similar FORTRAN structures that would be in a custom written program. The table computation commands are also conceptually easier to comprehend than the associated FORTRAN statements.

REFERENCES

1. Frankel, M.--"Software For Surveys--Are existing Packages Adequate for Valid Statistical Inference" Proc. Ninth Interface On Statistics and Computer Science - 1976

2. Levenson, Stephen B.--"Table Producing Language, Version 3.5, Users Guide", Bureau of Labor Statistics, Washington, D.C.

3. "Census Tabulation System-Aid" National Data Use and Access Laboratories

4. Williams, Elbert--"Generalized Tabulation System - User's Manual, Version 1.1" Bureau of the Census, Suitland, Maryland.

5. Francis I. Sherman, S. Heiberger--"Languages and Programs For Tabulating Data From Surveys" Proc. Ninth Interface On Statistics and Computer Science - 1976

6. Sedransk, J.--"Remarks on the Components of Compute Packages for Sample Surveys" Proc. Ninth Interface on Statistics and Computer Science - 1976

7. Cochran, W.--"Sampling Techniques" John Weley, New York - 1963

Least Squares Spline for Incorrectly-Posed Problems: Information Theoretic Approach

Kunio Tanabe
Institute of Statistical Mathematics, Tokyo, Japan
and
Brookhaven National Laboratory, Upton, N.Y. 11973

The use of the least squares splines for solving ill-posed (multi-colinear) problems is discussed. It is shown that choosing appropriate number of nodes, locations of nodes and degree of splines, we can control the smoothness of approximation in terms of spline functions, without resorting to Tikhonov's regularization (ridge regression) method which is designed to adjust the smoothness of the model functions to be fitted to data. It is also demonstrated that the Akaike's Information Criterion can successfully be applied to select the appropriate number of free parameters in the 'spline regression' models in which the number of nodes, locations of nodes and degree of splines are variable, based solely on data without using a subjective criterion such as significance levels in the statistical hypothesis testing.

The recovery of some definite structures from noisy data forms an important subject in experimental sciences. The 'inverse problem' in physical sciences are typical sources of this type of problems. These problems are inherently ill-conditioned in the sense that small disturbances in data cause a disastrous effect on the direct estimate of the true structure, although sometimes the problems are considered improperly formulated. They are called 'incorrectly-posed problems' and extensive efforts have been devoted to them in the literature for a reasonable estimate of the structures.

The author re-examined the subject from the information theoretical point of view which was originally introduced by Akaike in his paper for fitting statistical models. Making use of the Kullback-Leibler information quantity and the asymptotic theory of likelihood ratio statistics, he proposed a new statistic based on what we call the Akaike's information criterion(AIC),

$$\text{AIC} = -2\log_e(\text{maximum likelihood of a model with respect to given data}) + (\text{degree of freedom of the model}),$$

and the model chosen is the one with the smallest AIC value among competetive models with varing degree of freedom. This procedure which we call minimum AIC estimation(MAICE) differs from the existing methods in that it does not require such a subjective criterion as the significance level in statistical hypothesis testing. Although its significance in the field of time series analysis was well demonstrated in Akaike's papers [1], apparently its potential use in other areas such as pattern recognition is not yet fully recognized. In this note we show its applicability to the resolution of noisy data by treating two typical problems.

The first problem is the regularization of a noisy ill-conditioned system of linear equations,

$$Ax = b,$$

whose coefficient matrix and the right hand side constant vector are corrupted by random noise. The problem is closely associated with the numerical solution of the Fredholm integral equation of the first kind,

$$\int K(s,t)x(t)dt = b(s),$$

which arises in many branches of physical sciences,

typically in the experimental sciences where phsycal data are measured by indirect sensing devices.

As was well demonstrated by Phillips [6], direct application of usual numerical procedures to the equation will result in an osscillatory solution which is oftennot in good agreement with the nature of the problem. This subject has been discussed extensively in the literature and various smoothing procedures have been developed. They are classified into two categories. The first one minimizes

$$\| b - Ax \|^2 + \rho \| x - x_a \|_W^2,$$

where $\| x \|^2 = x^*x$, $\| x \|_W^2 = x^*Wx$ and the positive definite matrix W, the vector x_a and the smoothing parameter ρ are determined from a priori information on the 'true' solution. The result thus obtained is given by

$$x = (A^*A + \rho W)^{-1}(A^*b + \rho W x_a).$$

Phillips [6] and Tikhonov [9] devised this method. Foster [3] and Strand & Westwater [7] gave statistical extensions and justifications of this approach to certain models in which disturbances in the coefficient matrix were not taken into account. Their method for determining the smoothing parameter is not entirely satisfactory since it depends on the rather precise knowledge of the covariance matrix of the statistical noise which is usually unknown. The other method uses the singular value decomposition to invert the equation by the least squares method for the rank deficient case. Golub & Saunders [4-a], Hanson [5] and Varah [11] used an ad hoc method for determining an effective rank of A. Both approaches will give satisfactory results so long as W, x_a and ρ or an effective rank of A are suitably chosen. However the choices depend heavily on subjective judgement of the investigator.

To solve this problem we introduced a statistical model which uses spline functions(Fig.1-3). Applying the MAICE procedure we can determine 'objectively' a reasonable approximate solution in which the gain in resolution is balanced against the amplification of the noise. Fig. 1-4~9 and 11~12 show the approximate (spline function) solutions of the 'backward heat equation for the system,

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \quad -\infty < x < +\infty.$$

This problem is reduced to solve the Fredholm integral equation of the first kind,

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{4\pi(t_2-t_1)}} \exp\left(-\frac{(x-y)^2}{4(t_2-t_1)}\right) u(y,t_1)\,dy = u(x,t_2), \qquad (0 < t_1 < t_2),$$

for a given $u(x,t_2)$. Approximating $u(x,t_1)$ in terms of a finite linear combination of B-splines and discretizing x-space by collocation, we obtain a linear algebraic equation. Fig. 1-1 shows temperature profiles $u(x,t)$ at the time $t_1 = 0.16$ and $t_2 = 1$. Fig. 1-2 shows an solution obtained by applying the usual least squares method directly to the discretized linear equation. Fig. 1-3~9 show the least squares (cubic spline function) solutions with varying number of node points. Fig. 1-10 shows the profile of the values of AIC vs. number of nodes of cubic spline functions. Fig. 11 and 12 show quadratic and linear spline approximate solutions which are respectively selected by MAICE from quadratic and linear spline least squares solutions with various node points. Recently Golub [4-b] and Wahba [12] proposed methods to determine ρ, using similar statistical idea.

The second problem is the reconstruction of a polynomial regression from noisy measurement. The difficulty of this type of smoothing is due to our ignorance of degree of the polynomial to be fitted to given data. If we keep the order of the regression polynomial too low, the solution is biased from the true structure. If we allow too many parameters in the regression polynomial, the solution is affected with noise. Thus the main problem is the choice of a suitable degree of the regression polynomial. This problem was treated by Anderson 2 from the statistical hypothesis testing point of view. But his method is not satisfactory since it depend on subjective judgement. Applying the MAICE procedure, we are free from the limitation. Fig. 2 shows the 2-dimensional regression polynomials with degrees 0 to 10, fitted to data(top right) which are generated by adding random numbers to the polynomial(top left),

$$(2x^3 - x)(y^2 - y)/2$$

evaluated at grid points in x-y plane.

The two problem sketched here will find direct application in the following important fields of science and engineering:

a) super-resolution of noisy patterns respectively obtained by radio telescopes, by mine detectors, by electron scanning beams and X-ray photographs,

b) deduction of temparature profiles in the

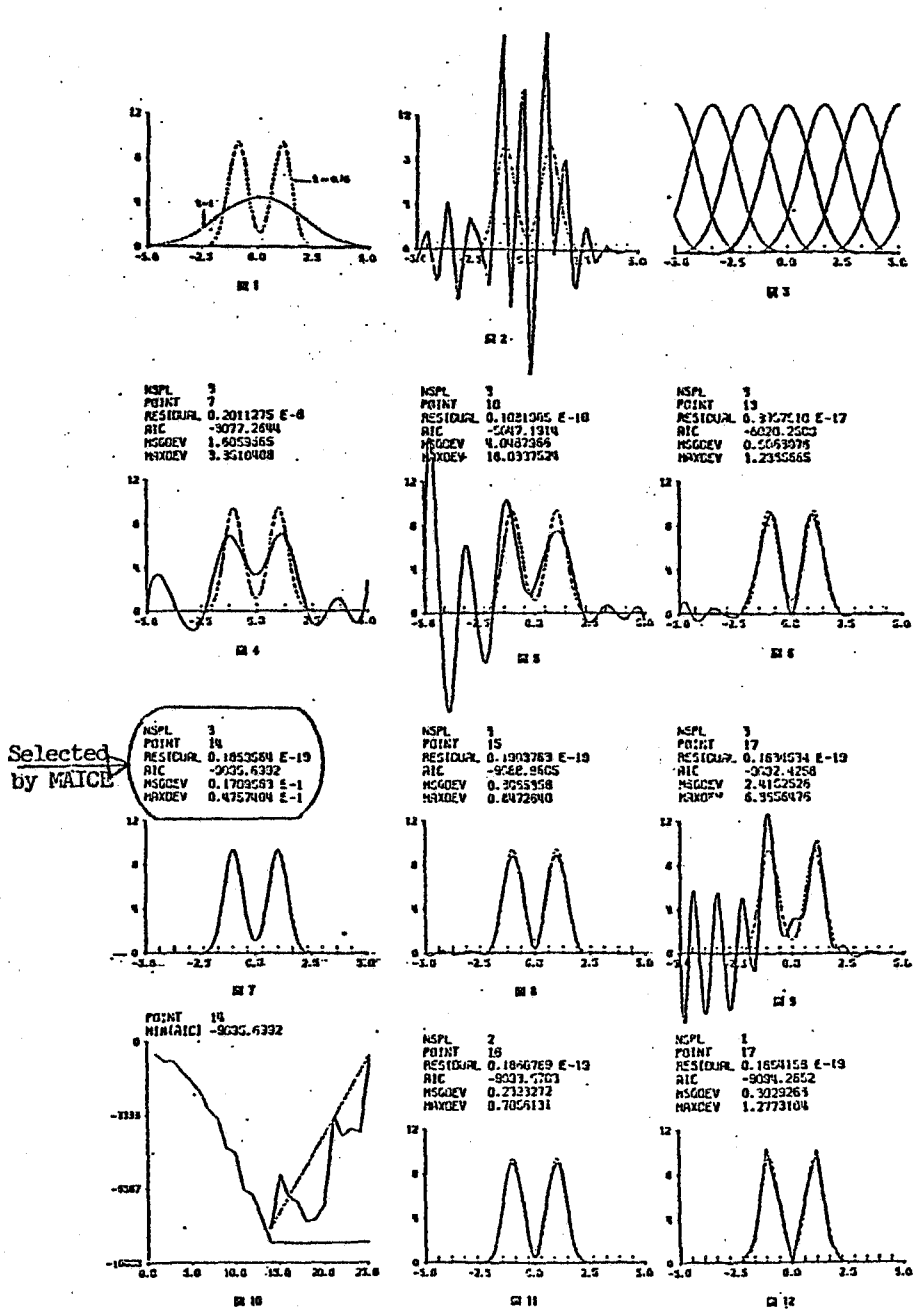


Fig.1 (K. Tanabe, Surikagaku, March 1976)

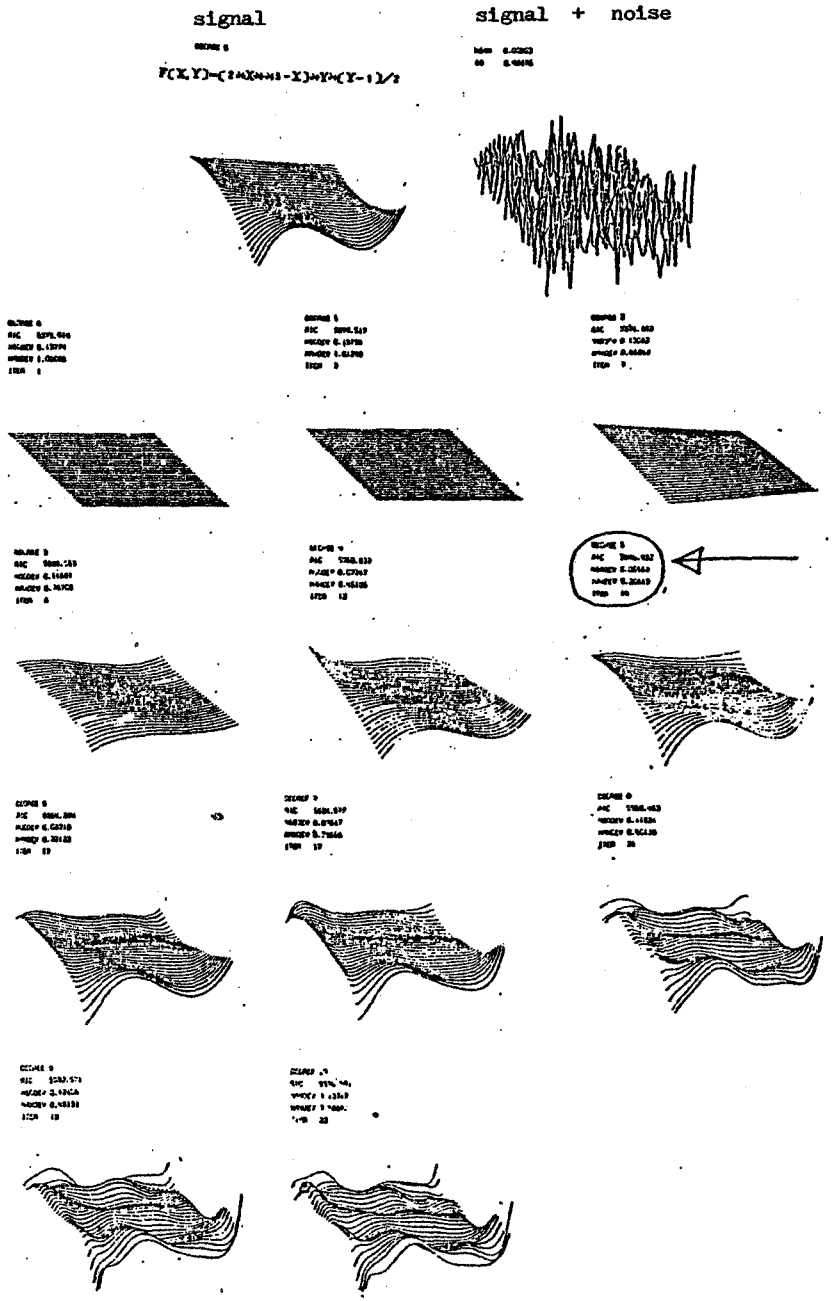


Fig.2 (K. Tanabe, 1974)

atmosphere by measuring the spectral distribution of infrared energy,

c) inference of aerosol size distribution from diffusional decay measurements of the total particle counts,

d) unsupervised estimation(learning without teacher) of mixture of density functions in pattern recognition,

e) numerical solution of ill-posed problems such as harmonic continuation and numerical inversion of Laplace transform.

REFERENCE

1. a) H. Akaike, "Information theory and an extension of the maximum likelihood principle, 2nd International Symposium on Information Theory, Problems of Control and Information Theory, B.N. petrov & F Csaki eds., 1973, pp. 267-281.
   b) H. Akaike, " Fitting autoregressive models for prediction" Ann. Inst. Statist. Math., vol. 21, pp.243-247, 1969.
   c) H. Akaike, "Statistical predictor identification" Ann. Inst. Statist. Math., vol. 22, pp. 203-217, 1970.
   d) H. Akaike, "A new look at the statistical model identification" IEEE Trans. on Automatic Control, vol. AC-19, pp.716-723, 1974.
2. T.W. Anderson, " The choice of the degree of a polynomial regression as a multiple decision problem I" Ann. Math. Statistics, vol.33, pp. 255-265, 1962.
3. M. Foster, "An application of the Wiener - Kolmogorov smoothing theory to matrix inversion" SIAM Appl. Math., vol. 9, pp.387-392, 1961.
4. a) G.H. Golub and M.A. Saunders, "Linear least squares and quadratic programming" Integer and Nonlinear Programming, pp.229-256, 1970.
   b) G.H. Golub, M. Heath and G. Wahba, "Cross-validation and optimum ridge regression" Stanford Computer Science report, 1976.
5. R.J. Hanson, 'Integral equation of immunology" CACM, vol.15, pp.883-890, 1972.
6. D.L. Phillips, "A technique for the numerical solution of certain integral equations of the first kind" JACM, vol.9, pp.84-97, 1962.
7. O.N. Strand and E.R. Westwater,"Minimum RMS estimation of the numerical solution of a Fredholm integral equation of the first kind" SIAM Numer. Anal., vol. 15, pp.287-295, 1968.
8. a) K. Tanabe, "Fitting regression curves and surfaces by Akaike's information criterion" Inst. Statist. Math. Res. Memo. no62, 1974.
   b) K. Tanabe, "Statistical regulariztion of a noisy ill-conditioned system of linear equations by Akaike's information criterion, Computation and Analysis, vol.6, pp.2-25, 1975.
   Above two papers were presented at the 1974 IEEE International Symposium on Information Theory held at Univ. NotreDame, Indiana, 1974.
   c) K. Tanabe, "Treatment of Statistical errors" (in Japanese), bit, vol.7, pp.113-125, 1975.
   d) K. Tanabe, "Statistical Approach to Incorrectly Posed problems"(in Japanese), Surikagaku(Math. Sciences), no. 153, pp.60-64, 1976.
9. A.N. Tikhonov, "The stability of algorithms for the solution of degenerate system of linear algebraic equations" USSR Computational Math. and Mathematical Physics, vol. 5, pp.181-188, 1965.
10. Twomey s., "The Application of numerical filtering to the solution of integral equations encountered in indirect sensing measurement" Jour. of Franklin Inst, vol.279, pp.95-109,1965.
11. J.M. Varah, " On the numerical solution of ill-conditioned linear systems with applications to ill-posed problems, SIAM Numer. Anal., vol. 2, pp. 257-267, 1973.
12. G. Wahba, "A survey of some smoothing problems and the method of generalized cross-validation for solving them" AMS Notes Short Course on Statistics, 1977.

A SHORT ALGORITHM TO TRANSFORM DISSIMILARITIES INTO DISTANCES

Hoang M. Thu
University of Cincinnati

ABSTRACT

In this paper, it is shown that given a set of dissimilarity data, one can find a parameter γ_{max} so that for all $0<\gamma\leq\gamma_{max}$ the power function $x\longrightarrow x^{\gamma}$ transforms the dissimilarities into distances. An algorithm is given with its FORTRAN IV module for computing γ_{max}.

I. INTRODUCTION

Many data analysis techniques involve the use of similarity or dissimilarity coeffi-cients considered respectively as measures of "proximity" or "distance" between objects in a finite set. The purpose is often

a) to represent this set of objects as a set of points in a Euclidean space so that their distances match with the dissimilarities between the corresponding objects

b) to find the best least-squares linear ad-justment to this set of points by a hyper-plane of small dimension.

Principal components, ordination in tax-onomy or ecology, and multidimensional scal-ing in psychology are examples of such meth-ods.

According to some authors, including Wil-liams and Dale(1965), Johnson(1968), Sokal and Sneath(1973), Clifford and Stephenson(1975), it is desirable that a dissimilarity co-efficient be a metric even when the assump-tion of the Euclidean structure of the embed-ding space is omitted.

Some computed dissimilarity coefficients and dissimilarities collected directly as data fail to satisfy the axioms of a metric. In these cases, transformations may help in "aid-ing in the analysis of data by bending the data nearer to the Procrustean bed of the as-sumptions underlying conventional analyses." (Tukey,1957)

This paper presents a simple algorithm to transform dissimilarity coefficients into distances.

II. POWER TRANSFORMATIONS OF DISSIMILARITY CO-EFFICIENTS INTO DISTANCES

Results similar to those proven below for the family of power transformations $x\longmapsto x^{\gamma}$ can be found for other families of trans-formations such as linear, exponential and logarithmic (or their composites).

First let us review some definitions. A positive function $d:I\times I\longrightarrow \mathbf{R}^{+}$ defined on the set of objects I may satisfy the following ax-ioms

(1) $d(i,j)=d(j,i)\geq 0$ for all $i,j\in I$
(2) $d(i,i)=0$ for all $i\in I$
(3) $d(i,j)=0 \Rightarrow i=j$ for all $i,j\in I$
(4) $d(i,j)+d(j,k)\geq d(i,k)$ for all $i,j,k\in I$

If d satisfies (1)-(2) it is called a dissimilarity coefficient

If d satisfies (1)-(4) it is called a distance (or metric).

Mathematicians believe that the triangle inequality axiom (4) is of central importance to the metric structure of a space. Therefore we focus our attention on dissimilarity coeffi-cients which violate this axiom, and choose to work with dissimilarity coefficients satis-fying axiom (3).

Denote the composite of $d:I\times I\longrightarrow \mathbb{R}^{+}$ and the power transformation $x\longrightarrow x^{\gamma}$ by d^{γ}. The problem is to find the set of all parameters γ such that d^{γ} satisfies (4) for all triples (i,j,k).

LEMMA. Given any three positive numbers a,b,c with $a=b+c$ then

(i) $a^{\gamma}<b^{\gamma}+c^{\gamma}$ for all $\gamma<1$
(ii) $a^{\gamma}>b^{\gamma}+c^{\gamma}$ for all $\gamma>1$

Proof: It is obvious.

COROLLARY. Given three positive numbers a,b,c the function $\gamma \rightarrow b^\gamma + c^\gamma - a^\gamma$ has at most one zero.

It has only one zero if, in addition, $b<a$ and $c<a$.

Proof: Suppose that $\gamma \longrightarrow b^\gamma + c^\gamma - a^\gamma$ has a zero γ_0. Applying the previous Lemma to a^{γ_0}, b^{γ_0} and c^{γ_0}, it is easily seen that this function has no other zero.

Now if, in addition, $b<a$ and $c<a$, i.e., $b/a<1$ and $c/a<1$, there is an γ_1 so that

$$b^{\gamma_1} + c^{\gamma_1} - a^{\gamma_1} = a^{\gamma_1}\left((b/a)^{\gamma_1} + (c/a)^{\gamma_1} - 1\right) < 0$$

since $(b/a)^\gamma$ and $(c/a)^\gamma$ tend to zero as $\gamma \rightarrow \infty$. On the other hand $b^\gamma + c^\gamma - a^\gamma \rightarrow 1$ as $\gamma \rightarrow 0$. Hence the continuous function $\gamma \rightarrow b^\gamma + c^\gamma - a^\gamma$ has a zero $\gamma_0 \in (0, \gamma_1)$.

PROPOSITION. Given a nonmetric dissimilarity coefficient, the set of all parameters γ so that d^γ is a metric is an interval $(0, \gamma_{max}]$.

Proof: Let a,b,c be three positive numbers. If $a \leq b$ or $a \leq c$ then $a^\gamma \leq b^\gamma + c^\gamma$, for all $\gamma > 0$. If $a>b$ and $a>c$ then by the Corollary the function $\gamma \rightarrow b^\gamma + c^\gamma - a^\gamma$ has exactly one zero, say $\gamma(a,b,c)$. Applying the Lemma, we have $a^\gamma \leq b^\gamma + c^\gamma$, for all $0 < \gamma \leq \gamma(a,b,c)$.

Therefore for any three positive numbers $a,b,c \in d(I \times I)$ with $a>b+c$, there is a positive number $\gamma(a,b,c)$ so that $a^\gamma < b^\gamma + c^\gamma$ for all $0 < \gamma \leq \gamma(a,b,c)$. The intersection of all such intervals

$$\bigcap_{a,b,c \in d(I \times I)} (0, \gamma(a,b,c)]$$

is the set of all parameters γ so that d^γ is a metric. It is an interval, say $(0, \gamma_{max}]$.

Now in choosing a specific γ, we recommend γ_{max}. Why?

First remark that in the Euclidean embedding problem a), for $\gamma=0$, the configuration associated with (I, d^γ) is equilateral and necessarily lies in a space of dimension n-1, (n being the number of objects in I). For $\gamma>0$, the smallest dimension of an embedding space can only decrease. We conjecture that within the class of power transformations of dissimilarity coefficients on I, the one associated with γ_{max} yields the space of smallest dimension. In any case γ_{max} is the point the most far remote from the trivial case $\gamma=0$.

III. ALGORITHM FOR POWER TRANSFORMATION OF DISSIMILARITY COEFFICIENTS INTO DISTANCES

(i) Find the triples (i,j,k) which violate the triangle inequality axiom (4)

(ii) Determine the corresponding $\gamma(i,j,k)$'s where $\gamma(i,j,k)$ is the largest number so that

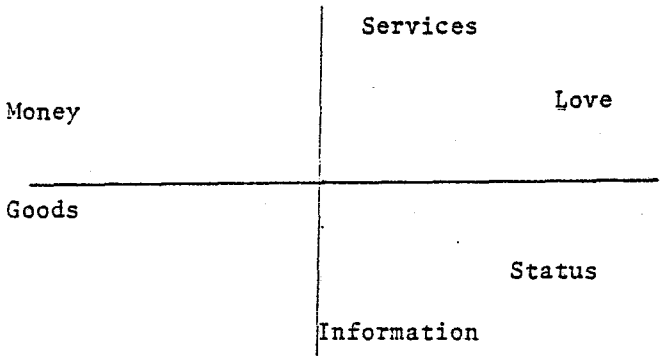
$$d^{\gamma(i,j,k)}$$

satisfies Axiom (4) for all permutations of the triple (i,j,k)

(iii) Approximate the smallest of all these $\gamma(i,j,k)$ by bisection.

IV. EXAMPLE

The example is taken from Foa's study of the exchange of material and psychological resources through interpersonal behavior (1971). Foa postulates a cyclic ordering of the six resource categories being investigated; love, status, information, money, goods and services. The similarity data are converted to dissimilarities which, in turn, are transformed into distances by power transformation with $\gamma = 0.556701E\ 00$ (0.56). The Figure shows the projection of the configuration onto the plane spanned by the two first eigenvectors produced by a multidimensional scaling procedure. The cyclic ordering is indeed recovered.



Acknowledgment. The author wishes to thank Prof. T.C. Chang and Prof. D. Gordon for helpful discussions.

REFERENCES

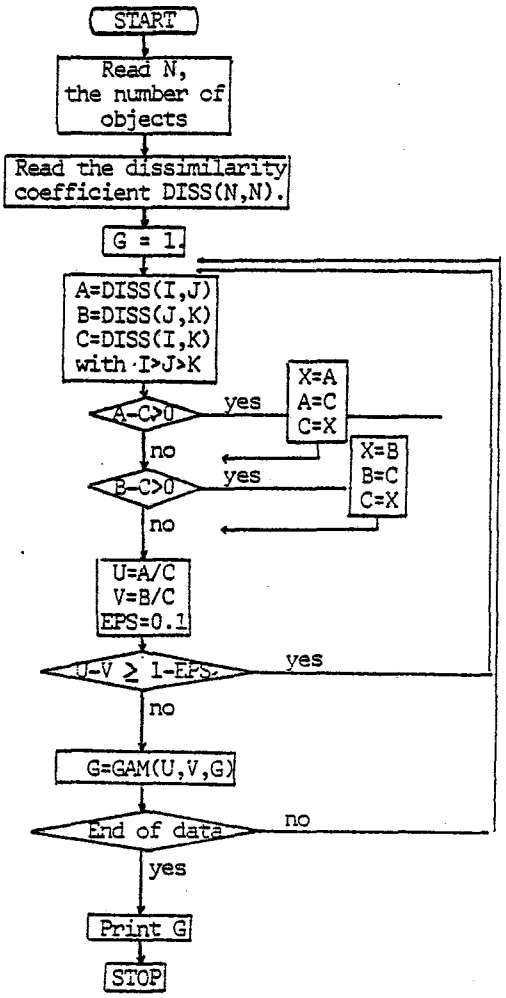
[1] Clifford H.T. & W. Stephenson (1975) An Introduction to Numerical Classification. NY, Ac. Press.

[2] Foa, U. (1971). Interpersonal and economic Resources. Science 171, 345-351.

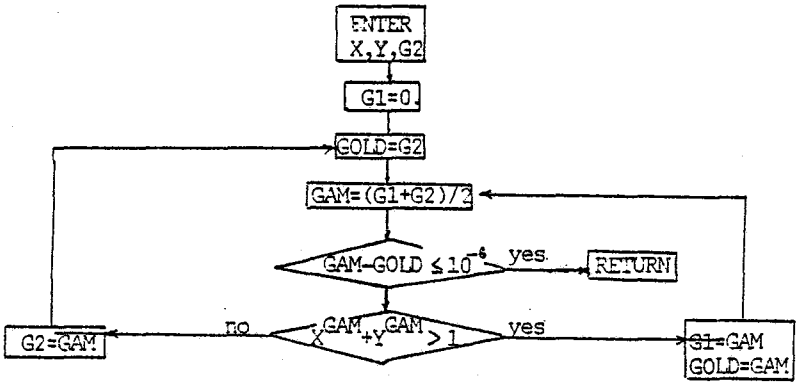
[3] Sneath, P.H. & R.R. Sokal (1973). Numerical Taxonomy. San Francisco, Freeman.

[4] Tukey, J.W. (1957). On the Comparative Anatomy of Transformations. Ann. Math. Stat. 28, 602-32.

[5] Williams, W.T. & M.B. Dale (1965). Fundamental Problems in Numerical Taxonomy. Advanced Bot. Res. 2, 35-68.



Function GAM(X,Y,G2)



APPENDIX.

THE FORTRAN IV CODE

```
C     POWER TRANSFORMATION OF
C     DISSIMILARITIES INTO DISTANCES

      DIMENSION DISS(40,40)
    1 FORMAT (I3)
    2 FORMAT (1X,40F3.0)
    3 FORMAT(1X,'GAMMA=  ',F15.7)
      READ 1,N
      DO 10 I=1,N
   10 READ 1,(DISS(I,J),J=1,N)
      G=1.
      N2=N-2
      N1=N-1

      DO 100 I=1,N2
      I1=I+1
      DO 100 J=I1,N1
      J1=J+1
      DO 100 K=J1,N
      A=DISS(I,J)
      B=DISS(J,K)
      C=DISS(I,K)
      IF(A.LE.C)GOTO 11
      X=A
      A=C
      C=X
   11 IF (B.LE.C)GOTO 22
      X=B
      B=C
      C=X
   22 U=A/C
      V=B/C
      IF(U**G+V**G.LE.1.-0.1) G=GAM(U,V,G)
  100 CONTINUE

      PRINT 3,G
      DO 55 I=1,N
      DO 57 J=1,N
      IF(DISS(I,J).EQ.0)GOTO 57
      DISS(I,J)=DISS(I,J)**G
   57 CONTINUE
   55 CONTINUE



      FUNCTION GAM(X,Y,G2)
      G1=0.
    3 GOLD=G2
    4 GAM=(G1+G2)*0.5
      IF(ABS(GAM-GOLD).LT.0.0001) RETURN
      IF (X**GAM+Y**GAM.GT.1) GOTO 1
      G2=GAM
      GOTO 3
    1 G1=GAM
      GOLD=GAM
      GOTO 4
      END
```

A NOTATION FOR SPECIFYING CONTINGENT EFFECTS IN ANALYSIS-OF-VARIANCE DESIGNS

Ervin H. Young

Institute for Research in Social Science
University of North Carolina at Chapel Hill

This paper defines contingent effects in multi-way analysis-of-variance designs, presents a notation for specifying such effects, provides an example of the notation, and discusses its advantages. The principal advantage claimed for the notation is its transparency.

Interaction effects, when they cannot be neglected, give rise to conceptual difficulties in interpreting the results of the analysis of variance. When two or more variables participate in an interaction, it may not be possible to make a global statement about the effect of any of them on the response variable, for the effect of each of them may depend on the combination of the levels of all the others. To get beyond the significant interaction, to find out how the effect of one factor depends on the level of the others, one needs to recast the model, using effects not found in the saturated fully-crossed design. I assert here, with only the justification of a single example, that the set of all contingent effects, as defined below, provides a sufficient basis for this recasting of the model, while the set of all nested effects does not. I go on to present a notation for specifying contingent effects, give an example of the notation, and discuss its advantages.

Restricted and Contingent Effects Defined

To define contingent effects, it is helpful to be able to refer to crossed and nested effects. A crossed effect is produced by one column of the saturated fully crossed design matrix, which is constructed by taking the direct matrix product of the indicator matrices for each factor in the design, in the same order as that employed in constructing the design table. The design matrix column for a restricted effect is obtained from the column for the corresponding crossed effect by setting to zero all elements that do not correspond to some proper subset of the cells of the design table; such an effect will be said to be restricted to that subset of the design. The cross-classification of a proper subset of the factors will be called a marginal design table. A nested effect may then be defined as a crossed effect restricted to one cell of the marginal design table for a non-trivial subset of the factors that do not participate in the crossed effect. A contingent effect may be defined as a crossed effect restricted to any proper subset of the cells of the marginal design table for a non-trivial subset of the factors that do not participate in the crossed effect. Clearly, for any non-trivial design, the set of all contingent effects is a superset of the set of all nested effects.

A Familiar Syntax for Generating Unrestricted Effects

Various simple devices are widely employed for naming the main and interaction effects in statements of analysis-of-variance models. Canned regression programs that permit only main-effects models usually use the simplest syntax, a list of the symbols for the regressors, with succeeding symbols in the list separated by blanks, as in the SPSS procedure REGRESSION. Names for interaction effects may be constructed by separating the symbols for the participating factors with asterisks, as in the MODEL statement for SAS procedures ANOVA and GLM. Alternatively, one might choose to enclose the name of each main or interaction effect in parentheses; for example, see Goodman (1970). For maximum emphasis, I have chosen to combine both of these conventions, using parentheses to surround the names of effects and asterisks to separate the names of factors participating in an interaction. For example, the model that includes the main effects of factors A, B, and C, and the interaction of B and C, is specified as

MODEL = (A)(B)(C)(B*C) /

with the slash terminating the model specification.

Hierarchical versus Non-hierarchical Specification

Goodman (1970) defines a hierarchical model as a subset of the fully-crossed design which, for every

interaction effect that it includes, also includes the main effects of the factors that participate in that interaction, and the interaction of every proper subset of them. This motivates the definition of a hierarchical specification of a model as one which, for each interaction that it explicitly includes, generates also the main effects and lower-order interaction effects needed to make the model hierarchical. A model specification that does not provide this service is naturally called non-hierarchical. Thus a hierarchical model may be generated not only by a hierarchical specification, but also by a non-hierarchical specification (if the user wants to go to that trouble), while a non-hierarchical model msut be generated by a non-hierarchical specification. For example, the model specified non-hierarchically above as

MODEL = (A)(B)(C)(B*C) /

can be specified hierarchically as

MODEL(H) = (A)(B*C) /

where the parenthesized H after MODEL indicates a hierarchical specification.

Extending the Syntax to Contingent Effects

A contingent effect was defined above as one derived from a crossed effect by restricting the latter to a proper subset of the cells of the marginal design table for some proper subset of the factors that do not participate in the crossed effect. Any such restriction may be specified by a statement whose truth characterizes only those cells to which the effect is restricted. A sufficient set of statements for specifying any such restriction is the set of statements of the form

arithmetic relation logical operator
arithmetic relation logical operator
. . .
logical operator arithmetic relation

where an arithmetic relation has the form

variable name relational operator value

and where logical operators are chosen from the set {AND,OR}, relational operators are chosen from the set {EQ,NE,GE,GT,LE,LT}, and the "value" of a factor is interpreted as the ordinal number of one of its levels.

For an example, consider again the three-factor model. If one wishes to include the main effects of A and B, but restrict the main effect of C to those cells for which either A or B are at their second levels, one may specify

MODEL = (A)(B)(C WHEN (A EQ 2 OR B EQ 2)) / .

Implementing the Notation

Continuing with the example just given, Table 1 illustrates the implementation of the notation. In Table 1, the first three columns show the stub of the design table for a design with three dichotomous factors A, B, and C. Columns 4 through 6 show the classical analysis-of-variance design matrix columns for the main effects (A), (B), and (C) respectively. The next two columns display the truth table for the proposition (A EQ 2 OR B EQ 2), in both truth-value notation (column 7) and numeric notation (column 8). The last column shows the design matrix column for the contingent effect (C WHEN (A EQ 2 OR B EQ 2)), which was obtained by a dyadic (element by element) multiplication of column 6 by column 8.

Nested Effects

A complete set of nested effects is defined here as a set of effects produced by restricting some

Table 1: An Implementation Example

1	2	3	4	5	6	7	8	9
A	B	C	(A)	(B)	(C)	(A EQ 2 OR B EQ 2)		(C WHEN (A EQ 2 OR B EQ 2))
1	1	1	-1	-1	-1	F	0	0
1	1	2	-1	-1	1	F	0	0
1	2	1	-1	1	-1	T	1	-1
1	2	2	-1	1	1	T	1	1
2	1	1	1	-1	-1	T	1	-1
2	1	2	1	-1	1	T	1	1
2	2	1	1	1	-1	T	1	-1
2	2	2	1	1	1	T	1	1

crossed effect to each cell in turn of the marginal design table of some subset of the factors that do not participate in the crossed effect. Thus, where B and C are dichotomies, the effects

(A WHEN (B EQ 1)) (A WHEN (B EQ 2))

form a complete set, as do the effects

(A WHEN (B EQ 1 AND C EQ 1))
(A WHEN (B EQ 2 AND C EQ 1))
(A WHEN (B EQ 1 AND C EQ 2))
(A WHEN (B EQ 2 AND C EQ 2)),

but any proper subset of either set is incomplete. Since it is often convenient to have a shorthand notation for a complete set of nested effects, I have provided one:

MODEL = (A WITHIN (B))

and

MODEL = (A WITHIN (B*C))

for the two examples just given.

Comparison of this Notation with Direct Matrix Input

The example shown here compares the notation presented above with the input to a program that requires direct matrix input of design matrices. The syntax described above is implemented in a forthcoming SPSS procedure for fitting models to categorical data (see Young, 1978). The comparison program is GENCAT, a program for fitting models to categorical data using weighted least squares (see Landis, et al., 1976). The test data were analyzed by Giles, et al., (1976) to compare racial prejudice with social class prejudice with respect to the strength of their effects on the probability of parents' engaging in some overt form of protest against public school desegregation. Figure 2 displays the SPSS statements necessary to replicate the three models for which Giles, et al., report results. Figure 1 shows the same three models defined in input to GENCAT. These inputs have been submitted to their respective computer programs, and have produced results in substantial agreement with each other and (with one minor exception) with those of Giles, Gatlin, and Cataldo (1976).

The principal advantage of the notation presented here is its transparency. It minimizes the amount of information that the user must acquire to use the program, over and above that which he requires for an understanding of the program's output. The user who understands what a main effect parameter may mean, and

Figure 1: GENCAT Instructions to Replicate Results of Giles, et al. (1976).

```
    7     1    32          1       (32F2.0)       SATURATED FULLY CROSSD
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1
 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1
 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1
 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 - 0 0 0 0 0 0 0 0 0 0 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
    8     1     1     0     0     0   (36F2.0)
 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
```

Figure 1, continued.

```
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 00 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
    8     1     1     0     0     0   (36F2.0)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
    7     1    10          1       (32F2.0)       R'ISM & C'ISM IN I*E
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
```

Figure 1, concluded.

```
0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0
   8    1    1    0    0    0  (36F2.0)
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   8    1    1    0    0    0  (36F2.0)
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   8    1    1    0    0    0  (36F2.0)
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   8    1    1    0    0    0  (36F2.0)
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   8    1    1    0    0    0  (36F2.0)
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   8    1    1    0    0    0  (36F2.0)
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   8    1    1    0    0    0  (36F2.0)
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   8    1    1    0    0    0  (36F2.0)
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   8    1    1    0    0    0  (36F2.0)
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   7    1    5         1         (32F2.0)          FINAL MODEL
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0
   8    1    1    0    0    0  (36F2.0)          %BLACK MAIN EFFECT
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   8    1    1    0    0    0  (36F2.0)          R'ISM WHEN INC=ED=LO
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   8    1    1    0    0    0  (36F2.0)          R'ISM WHEN INC NOT= ED
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   8    1    1    0    0    0  (36F2.0)          C'ISM WHEN INC=ED=HI
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 2: SPSS Instructions to Replicate Results of Giles, et al. (1976).

```
RUN NAME        RACE & CLASS PREJUDICE,GILES ET AL.,ASR41(4/76),280-28
VARIABLE LIST   PROTESTR %BLACK RACISM CLASSISM INCOME EDUCATION
INPUT MEDIUM    TAPE
N OF CASES      UNKNOWN
ALLOCATE        TRANSPACE = 1000
VAR LABELS      PROTESTR DISAPP DESEG'N HNDLG & SAID SO TO SCH OFFICLS,
                %BLACK CHANGE IN KIDS' SCH % BLACK FROM 71-72 TO 72-73,
                RACISM SUMMED RACIAL PREJUDICE SCORE /
                CLASSISM SUMMED CLASS PREJUDICE SCORE /
VALUE LABELS    PROTESTR (1)NO (2)YES /
                %BLACK (1)OTHER (2)<30%71-2,>30%72-3 /
                RACISM CLASSISM (1)BELOW MEDIAN (2)ABOVE MEDIAN /
                INCOME (1)<=$14999 (2)$15000+ /
                EDUCATION (1)<=HS DIPLOMA (2)BEYOND HS /
CROSSTABS       VARIABLES = PROTESTR TO EDUCATION (1,2) /
                TABLES = PROTESTR BY %BLACK BY RACISM BY CLASSISM
                  BY INCOME BY EDUCATION /
                  ESTIMATE = WLS / RESPONSE = LINEAR /
                    MODEL(H) = SATURATED /
                    MODEL =  (MEAN) (%BLACK)
                      (RACISM WITHIN (INCOME*EDUCATION))
                      (CLASSISM WITHIN (INCOME*EDUCATION)) /
                    MODEL = (MEAN) (%BLACK)
                      (RACISM WHEN (INCOME EQ 1 AND EDUCATION EQ 1))
                      (RACISM WHEN ((INCOME EQ 1 AND EDUCATION EQ 2) OR
                        (INCOME EQ 2 AND EDUCATION EQ 1)))
                      (CLASSISM WHEN (INCOME EQ 2 AND EDUCATION EQ 2))/
OPTIONS         11
STATISTICS      16
```

who understands how a large interaction parameter renders the main effects of participating factors meaningless, will be able to employ the syntax presented here to analyze the dependency of some factors' effects on the levels of other factors, using only the rudimentary formal logic that he presumably already knows. As an ancillary benefit, since the specification of a contingent effect parameter also states its interpretation, the specification syntax itself can be used to label the parameter on the program output.

It might be thought from a cursory inspection of Figures 1 and 2 that another major advantage of this notation is brevity. Examining the Figures more closely, however, one will notice that the last of the three models, the only one that requires for its specification the notation for contingent effects that is novel to the work reported here, requires 168 non-blank keystrokes is SPSS versus 305 in GENCAT. On the other hand, the second model, which includes two complete sets of nested effects, requires 87 non-blank keystrokes in SPSS vs. 640 in GENCAT, and the saturated fully-crossed model requires 49 keystrokes in SPSS vs. 2109 in GENCAT. Thus, in specifying the contingent effects model, the user has gained an advantage of less than 2 to 1 with respect to physical effort, whereas the advantage is of the order of 7 to 1 for the nested model and 40 to 1 for the saturated model. For this reason, brevity is not claimed as the major advantage of the notation for contingent effects.

One should also note GENCAT's versatility with respect to the variety of models that can be fit. The syntax presented here is not intended to capture the full generality of GENCAT.

Summary

This paper defines contingent effects in the analysis of variance, and presents a notation for specifying such effects in computer program input. The principal advantage of the notation is its transparency. It provides a self-interpreting mnemonic syntax that even a relatively unsophisticated user can employ to carry a traditional analysis of variance to its conceptual completion.

References

Barr, Anthony J., James H. Goodnight, John P. Sall, and Jane T. Helwig (1976) A User's Guide to SAS 76. Raleigh, N. C.: SAS Institute, Inc.

Giles, Micheal W., Douglas S. Gatlin, and Everett F. Cataldo (1976) "Racial and class prejudice: their relative effects on protest against school desegregation." American Sociological Review 41(April): 280-288.

Goodman, Leo A. (1970) "The multivariate analysis of qualitative data: interactions among multiple classifications." Journal of the American Statistical Association 65(March): 226-256.

Landis, J. Richard, William M. Stanish, Jean L. Freeman, and Gary G. Koch (1976) A Computer Program for the Generalized Chi-Square Analysis of Categorical Data Using Weighted Least Squares. Technical Report No. 8, Department of Biostatistics, University of North Carolina at Chapel Hill.

Nie, Norman H., C Hadlai Hull, Jean G. Jenkins, Karin Steinbrenner, and Dale H. Bent (1975) SPSS: Statistical Package for the Social Sciences (2nd Edition). New York: McGraw-Hill.

Young, Ervin H. (1978) An SPSS Procedure for Fitting Models to Categorical Data. Technical Paper No. 4, Institute for Research in Social Science, University of North Carolina at Chapel Hill.

CONTRIBUTIONS BY ABSTRACT

EDIT AND IMPUTATION PROCEDURES AT STATISTICS CANADA

Len Baniuk, J. H. Johnson, G. Sande
Statistics Canada

Displays concerning three aspects of the editing and imputation of statistical data will be made. (1) The methods of Fellegi and Holt for coded data which has been used for census data processing. (2) Imputation of survey data to an administrative data frame which has been used in the place of ratio estimation methods. (3) A numerical data edit and imputation test system which is under development. Each of these uses a hot deck to supply donated values in the imputation. For numerical data, the matching is done by examining nearest neighbours. The use of imputation methods, rather than ratio estimation, yields consistent subpopulation values and uses the frame data to compensate for survey biases. Methods from operations research are used to identify minimal edits and the fields to impute in the numerical data test system.

A FAST ALGORITHM FOR ESTIMATING THE NUMBER OF POLYNEUTRONS DETECTED IN THE PRESENCE OF POISSON NOISE

W. A. Beyer
Los Alamos Scientific Lab., Los Alamos, NM

C. Qualls

Department of Mathematics and Statistics
University of New Mexico, Albuquerque, NM

An experiment was conducted by Turkevich et al ['Search for Particle-Bound Polyneutron Systems', Phys. Rev. Lett. 38, 1129-1131 (1977)] to measure the number of polyneutrons produced in the reaction: 800 MeV protrons + target → polyneutrons + fragments. The detection of possible decays of polyneutrons is complicated by similar decays arising from noise (an unknown amount of contamination is introduced). Maximum likelihood estimates of the number of polyneutrons present can be obtained in an implicit manner by the use of a computer. However, in order to obtain these estimates within a reasonable amount of computer time, a probability generating function idea is applied resulting in a fast computing algorithm. A goodness of fit result is also obtained. (Both the decays of polyneutrons and the decays of contaminates are modeled by pure death processes with known Poisson decay rates.)

RUMMAGE - A DATA ANALYSIS SYSTEM

G. Rex Bryce, Del T. Scott and Melvin W. Carter
Brigham Young University, Provo, Utah

This paper outlines unique approach to the analysis of nonorthogonal data sets and it's implementation in the Rummage package. Starting with the simple cell means model a straight forward reparameterization yields a model which is analogous to the usual overparameterized model but which retains the full rank properties of the cell means model. Using English like commands, the testing of most hypotheses of interest in a designed experiment and the estimation of marginal means under various model assumptions are demonstrated. This includes exact methods in the case of fixed models and approximate methods for mixed and random models.

One of the most important features of the Rummage package is the inclusion of expected mean square coefficients with each Anova table. This additional information allows one to ascertain which factors in the model are contaminating each mean square. In mixed and random models this information is used directly to indicate appropriate test ratios. In the fixed model the terminology "expected mean square coefficient" is actually a misnomer. However, the coefficients are still extremely useful to show the degree of contamination with other marginal means in each means square.

A PREVIEW OF P-STAT 78

Roald Buhler and Shirrell Buhler
Princeton University Computer Center
87 Prospect Ave.
Princeton, N. J., USA

P-STAT 78, a new version of P-STAT now being tested, has substantial changes in interactive use, cross-tabulation and data display.

Interactive support now allows automatic screen holding. There is also an increased amount of prompting. HELP has been implemented in all new commands and is gradually being incorporated into the rest of the system.

Cross-tabulation changes include: (1) nested tables, such as SEX WITHIN AGE BY URBAN.RURAL WITHIN EDUCATION, which provides a two way table of those four

variables - the levels of SEX appear within each level of AGE, etc. and; (2) an interactive dialog mode that permits a table to be defined, displayed, some of its levels combined, re-displayed and optionally converted into a P-STAT file.

LIST, a new data display program, prints a file as neatly as possible while using a minimum page or screen width. Its features include: (1) deciding automatically how many decimal places a variable has and only printing that many; (2) printing score labels instead of values whenever possible; (3) providing optional sub-group headings within the listing, and; (4) parsing of labels to be readable in various column widths.

Examples of output from these programs will be shown.

GRAPHICS FOR SEASONAL TIME SERIES ANALYSIS

William S. Cleveland, Douglas M. Dunn, and Irma J. Terpenning
Bell Laboratories, Murray Hill, N.J.

The analysis of seasonal time series often presents problems because of the two-way structure in the data. Looking across months (i.e., observations within a cycle) one is interested in the long-term (trend) properties of the data. Analysis of the data for a given month (i.e., across cycles) provides insight into the form of the seasonal behavior. Seasonal adjustment algorithms can often aid in the analysis of seasonal data by decomposing it into trend, seasonal, and irregular components.

Graphical methods are presented which both summarize and display the within and across months structure of seasonal data. Alternative plotting procedures as well as different types of displays are developed and contrasted. Displays are presented to aid in the evaluation of a seasonal adjustment decomposition with attention focused on detecting "leakage" of one component (e.g., seasonal) into another (e.g., trend).

Various sets of seasonal macroeconomic data are used to illustrate the methods and displays. All procedures are implemented using portable graphics software.

MODULAR PROGRAMMING IN NUMERICAL AND STATISTICAL ANALYSIS

W. H. Connelly
Babcock and Wilcox Power Generation Group
Nuclear Power Generation Division
P. O. Box 1260, Lynchburg, Virginia 24505

General requirements for constructing a useful system of computer programs for performing numerical and statistical analyses are given. Although the overall system becomes "computer-dependent", the major portions of the calculations are shown to be based on computer-independent, or portable, FORTRAN subroutines. The modular structure utilizing overlays, dynamic storage allocation and free format input, is illustrated by construction on a CDC-7600. The desirable characteristics which result are viewed from a programmer's viewpoint and from a user's viewpoint. Replacement of outdated algorithms, expansion of existing capabilities and addition of new ones are discussed. Examples are taken from numerical and statistical techniques such as least squares and spline interpolation. The one-to-one correspondence possible when documenting input and output requirements of the modules is illustrated. The use of such a system in construction, testing and verifying large scale engineering and other scientific application programs is also illustrated.

CONFIDENTIALITY PROCEDURES IN STATISTICAL AGENCIES

L. H. Cox
Statistical Research Division
U.S. Bureau of the Census

G. Sande
Business Survey Methods Division
Statistics Canada

Statistical agencies must insure that none of their published data can be related to identifiable individuals. Differing methods are used to achieve this end in the cases of (1) microdata files for public use, (2) tabulations from population census and (3) tabulations from economic censuses. The various techniques used include identifier suppression, detail suppression, data perturbation,

random rounding, category roll up, cell and table suppression, range publication and delayed publication. These actions may be carried out directly by the statistical agencies or required by them of the users of their confidential data. In this poster we seek to illustrate the techniques used and some analysis of their impact. Many interesting problems remain concerning the impact of statistical analysis on the confidentiality of data collected by statistical agencies.

U R W A S
UNIVERSITY OF ROCHESTER WEIGHTED ANOVA SYSTEM

Michael L. Davidson and Jerome D. Toporek, University of Rochester

The question of why different Analysis of Variance programs give different results for the same problem has been asked and discussed many times in recent years. The concept of cell weighting has been presented as a unified explanation of this problem. URWAS, the University of Rochester Weighted ANOVA System is a generalized ANOVA program which provides the user with the ability to perform tests of hypotheses and estimation of parameters based on explicitly requested cell weighting, rather than on program imposed sample size related cell weighting.

URWAS is now available for distribution. The current version of the program along with associated documentation is presented. Information on distribution and implementation is also given.

ESTIMATING LATENT SCORE REGRESSIONS WHEN MEASUREMENT ERROR VARIANCES ARE KNOWN

Noel Dunivant, Ph.D., Assistant Professor
Psychology Department, New York University
6 Washington Place, 7th Floor, New York, NY 10003

The purpose of this paper is to briefly review four statistical methods for estimating errors-in-variables models, to describe the properties of the estimators produced by these methods, and to compare several computing algorithms for determining the estimates. These methods share three characteristics: (1) they require a priori information about the variance structure of the measurement

errors; (2) they are appropriate for the multiple regression or analysis of covariance situations; and (3) errors in equations as well as errors in variables are permitted. The first method, developed by Stouffer (1936a,b), Koopmans (1937) and Lindley (1947), produces maximum likelihood estimators, and a likelihood ratio test can be derived for testing hypotheses about coefficients of the structural relations. Stroud (1968, 1972, 1974) used Wald's (1943) method of unrestricted maximum likelihood to devise an asymptotic chi square test of the hypothesis of equality of conditional true score means for two groups. Fuller (1977) and his associates have developed consistent estimators of the true score parameters from a covariance matrix which is constrained to be positive definite. In addition, the principle of k-class estimators is incorporated into the estimation equations to produce estimates in small samples which have smaller mean square error. Formulas for large sample confidence intervals are given which include information about the sampling distribution of the measurement error variances, if available. In the last procedure standard psychometric formulas are employed to estimate true scores for each person; then the usual regression analysis is performed (Porter & Chibucos, 1974). In the paper I demonstrate how each of the four methods can be computed using FORTRAN programs and statistical computing languages such as OMNITAB II and SPSS. Primary emphasis is devoted to a comparison of the performances of the following programs in terms of accuracy, speed, and ease of use: Fuller's SUPER CARP, Jöreskog's LISREL, SPSS, and OMNITAB II.

PATIENT PROFILING SYSTEMS THROUGH A MINICOMPUTER

Turkan K. Gardenier, Ph.D.
Columbia University and Teka Trends, Inc.

The paper will present a multi-module computer package with applications to health evaluation facilities in an outpatient clinic. The system has been designed to incorporate patients' demographic characteristics as well as to generate continuous indices of improvement over time. Statistical techniques designed by the authors to monitor changes in enzyme levels, physician's global evaluation of improvement and the co-occurrence of various health status indices are computerized to summarize the physical status of the patient.

Demonstrations will be provided of the case-by-case applications of the

profiling systems associated with charts and superimposed graphs of summary status over time. Characteristics of the system are reminiscent of the profiling depicted by SMA-12 summaries. They further provide diagnostic profiling of a patient's status by reference to correlations of status with probability of future outome.

A summary of system efficiency will be presented through trade-offs of the sort-merge algorithm used in this system with those of extended core space in a larger scale computer.

BMDP-77 GRAPHICAL DISPLAYS AND RECENT PROGRAM RELEASES

MaryAnn Hill
Health Sciences Computing Facility, UCLA, and
Brentwood VA Hospital Statistical Research Unit

The developers of BMDP-77 believe that graphical displays are an integral part of data analysis and the communication of results. Displays include, for example,

- a matrix of shaded symbols representing correlation size;
- side-by-side histograms of data for each cell in analysis of variance;
- side-by-side plots for four variables of cell means for multifactor data structures;
- plots of factor loadings, factor scores and canonical variable scores;
- multiway contingency tables;
- user-defined symbols to represent <u>additional</u> variables in bivariate scatter plots

There are many types of plots for residuals:

residuals	vs.	predicted values
squared residuals	vs.	predicted values
residuals	vs.	residuals of deleted cases
$y - \bar{y} + bx_i$	vs.	x_i
residuals	vs.	cumulative normal

In one program the residuals to be plotted can be standardized, weighted, or for deleted cases. A plot is also available for both the observed and the predicted value versus each independent variable.

New programs available in BMDP-77 include

P2F Two-Way Tables - Empty Cells and Departures from Independence

Analyzes a subset of cells in a frequency table or identifies the outlier cells that contribute most to a significant chi-square test.

P3F Multiway Frequency Tables -- The Log-Linear Model

Forms and fits a log linear model to multiway frequency tables.

P1L Life Tables and Survival Functions

PAM Description and Estimation of Missing Data

P9R All Possible Subsets Regression

With residual analysis.

PAR Derivative-free Nonlinear Regression

P3V General Mixed Model Analysis of Variance

With parameter estimation by maximum likelihood or by restricted maximum likelihood.

Copies of the BMDP-77 manual are available for inspection. It has been totally rewritten and includes a comprehensive description of the new programs listed above in addition to the 26 programs released in 1975. Current and past issues of BMD Communications are also available.

BMDP PROGRAMS UNDER DEVELOPMENT AND HSCF TECHNICAL REPORTS

MaryAnn Hill
Health Sciences Computing Facility, UCLA, and
Brentwood VA Hospital Statistical Research Unit

A Stepwise Logistic Regression program and a k-Means Clustering program are scheduled for release in late 1978 by BMDP. Output and documentation for these programs are displayed.

The Logistic Regression program proceeds in a stepwise manner to estimate the parameters of the linear logistic model. In this model the dependent variable is dichotomous (e.g., success or failure) and the independent variables are categorical or continuous. The program generates design variables for the categorical variables and their interactions. In the stepping process the design variables for each categorical variable (or interaction term) are considered as a set -- at each step, a continuous variable or one set of design variables is added to or removed from the model. The program estimates the parameters to maximize the

likelihood function. Several graphical displays and tabulations are available to augment the analysis.

The k-Means Clustering program defines initial cluster centers (seeds) and assigns each case to the closest center. The seed is redefined after all cases are allocated, and the process is repeated until there is no further change. A metric inequality is applied to the distances between the present cluster centers and the original centers and the distances between cases and the original seeds, greatly reducing the number of distance computations necessary. Usually, for each iteration, only distance to cluster centers that are close to the case need recomputation. Winsorization is used in computing cluster centers and variances -- differences exceeding a threshold are truncated to the threshold. Missing values are treated in distance calculations as if they are very large; they are ignored in computing centers and variances. Graphical displays and comparative statistics are available for deciding the number of clusters justified by the data.

A sampling of HSCF Technical Reports on display and available for discussion include

8. Annotated Computer Output for Factor Analysis Using Professor Jarvik's Smoking Questionnaire (1974) - James Frane and MaryAnn Hill.
23. Asymptotic Standard Errors and Their Sampling Behavior for Measures of Association and Correlation in the Two-Way Contingency Table: I. Testing the Null Hypothesis; II. Power and Confidence Limits (1976) - Morton B. Brown and Jacqueline Benedetti.
24. LSD: Linear System Function and Derivatives Subroutine (1976) - R. I. Jennrich and A. D. Thrall.
28. Randomization or Minimization in the Treatment Assignment of Patient Trials: Validity and Power (1977) - A. B. Forsythe and F. W. Stitt.
29. Dud, A Derivative-Free Algorithm for Nonlinear Least Squares (1977) - M. L. Ralston and R. I. Jennrich.
30. Robust Estimators of Location for Biomedical Applications (1977) - MaryAnn Hill.
32. Ridge Regression, Bayes Estimation, Variance Components and the General Mixed Model (1977) - R. I. Jennrich.
34. Letting BMD08V Tell Us Something About Randomized Blocks, Repeated Measures and Split Plots (1977) - R. I. Jennrich.
35. Methods in BMDP for Dealing with Ill-Conditioned Data (Interface, 1978)

- James W. Frane.

36. Some Problems Faced in Making a Variance Component Algorithm into a General Mixed Model Program (Interface, 1978) - R. I. Jennrich and P. F. Sampson.

37. A Computer Program for Model Building with Stepwise Logistic Regression (Interface, 1978) - L. Engelman.

A COMPARISON OF THE CALCULATION OF SAMPLE MOMENTS IN BMDP, SAS AND SPSS

F. Kent Kuiper, Senior Statistician
Boeing Computer Services Company
Seattle, Washington

The computational accuracy of BMDP, SAS and SPSS are compared in their calculation of the sample mean, standard deviation and correlation. Identical experiments were run on all three packages, and in the case of BMDP and SPSS, they were run on both IBM and CDC computers. The results are compared to similar results in a previously published article on the IBM versions of BMDP and SPSS, which showed inaccuracies in the computation of the correlation coefficients using Pearson Corr in SPSS. These inaccuracies appear to be corrected in the latest versions of this package.

DATA ANALYTIC DISPLAYS FOR RIDGE REGRESSION

R. L. Obenchain
Bell Laboratories
Holmdel, New Jersey

Ridge regression reveals the possible effects of ill-conditioning among regressor variables upon the relative magnitudes and signs of fitted regression coefficients. But, in addition to the ridge trace plot of coefficients, one also needs to examine various summary statistics and estimated mean-squared-error traces to develop a well informed opinion about which ridge estimators are "better" than the least squares solution. This approach is illustrated by output from a Fortran program, RELAXR, that implements techniques from my published and unpublished papers.

A COMPARISON OF FOUR ALGORITHMS FOR COMPUTING EXPECTATIONS, VARIANCES AND COVARIANCES OF MEAN SQUARES

F. M. Speed, A. T. Coleman, L. W. Murray
Mississippi State University

This paper considers the following algorithms: i) the procedure in Rummage; ii) VARCOMP - the SAS procedure; iii) "SYNTHESIS" - Hartley's Method; and iv) the procedure suggested by Hocking and Speed. (Biometrics, 1974). Points of comparison include i) which algorithms calculate which moments, ii) what type models are allowed by each algorithm, iii) core size requirements of each algorithm.

In addition, certain computational techniques are discussed. These include i) the use of the Cholesky decomposition in finding the moments of the mean squares and ii) a method of efficiently generating the covariance matrix.

Finally, there is a discussion of the lack of uniformity of expected mean square tables.