STATISTICAL COMPUTATIONS--A SURVEY

E. J. Carney

Biometrics Unit, Cornell University
Ithaca, New York

BU-494-M

January, 1974

Abstract

A brief survey was made of activities in which a combination of knowledge of statistics and computer science is beneficial. Included are statistical data processing, Monte Carlo studies, symbolic computations, computer simulation, evaluation of operating systems, analysis of algorithms, program testing, and artificial intelligence. An attempt is made to point out the need for the application of both statistics and computer science in these areas.

This discussion will attempt to survey activities which may be grouped under the heading "statistical computations". Of course no one will agree what activities should be included in that category. The criterion used here will be that the activity should be included if its practice requires knowledge of statistics and computer science. This approach ignores such questions as "What is statistics?", "What is computer science?", since the answers to these questions are well known.

There does seem to be some parallel between the fields of computer science and statistics. In statistics there is a basic problem of determining what things can be apprehended through systematic observation, and then many subsidiary questions about how this may best be done. In computer science there

is a somewhat similar basic question of what functions can be computed, and then the many subsidiary how to do it questions. In both fields the basic questions are often left unanswered and cause little trouble unless we try to estimate something which is not estimable, or to compute something for which no algorithm can exist. The subsidiary questions provide adequate exercise for most of us.

An obvious meeting place for statistics and computer science is in the processing of data. Statistics will be concerned with deciding what functions of the data to compute and computer science with the development of algorithms to compute them. Some may say that these are separate problems and should be dealt with separately, but my experience indicates that this is not the case. Perhaps some anecdotal evidence will illustrate the difficulty of dealing with these problems separately.

On one occasion, while visiting a local Rhode Island business establishment, I was approached by a colleague. After it was determined who was buying, he began to complain that our computer was too small. This surprised me because our computer had a large memory, and his problem was a rather simple one: To obtain a frequency distribution for several variables. The difficulty was that he had about 100,000 observations, and the computer program he was using required all the observations to be in memory at once. To obtain a frequency distribution it is only necessary to look at one observation at a time, and so his difficulty could be resolved by some slight modification of the program. It should be pointed out that this isn't quite as easy as it sounds, because changing the program to read a single observation, update the frequency table, and then go on to read the next observation might result in an excessive number of input operations, exchanging the problem of too much storage for one of too much time. Even so, an efficient program for his problem could be developed.

What does this rather routine encounter between consultant and consultee illustrate? First, we have the question: Why does he have 100,000 observations? This is more a statistical question than it is a computer science question, although those computer scientists who see statisticians as anachronisms might feel that it is just a common sense question. Given that the frequency distribution is needed, there are statistical questions relating to sampling, randomness etc. involved in gathering the data. There are also problems of getting the data into a form to be fed to the computer. As we have seen, there are decisions to be made with respect to the nature of the algorithm to be used. There is here an interaction between the observational environment and the computational environment and good design for handling this data requires knowledge of both.

On another occasion a fairly prominent statistician complained to me that he had developed a good statistical method but that none of the programmers at his University had brains enough to program it. He may have been correct in his assessment, but there is some possibility that his proposed method was not good enough because it was numerically unstable, or because it required such a large number of operations that it was not economically feasible. Proving that a given sequence is convergent in the real numbers does not mean that an algorithm with approximates that sequence using numbers with a finite number of digits will be convergent in the sense that after a few hundred thousand arithmetic operations (usually all one can afford for the answer) it will give a sufficiently close approximation to the limit of the sequence.

A somewhat similar case comes to mind in which a statistician found an integral expression for the distribution of a proposed test statistic. Because she wanted to make some power tables, the statistician worked very hard to find an analytic expression for this integral, and was able to get a series expansion.

She then found a programmer to program this expansion, but it took a great deal of computer time to compute even a single point. The programmer then took the expression to a numerical analyst who got back to the original integral, which, it turned out, was easily integrated numerically.

One could go on with such examples, but these three should be enough to document my prejudice in this matter. What is their point? That a statistician needs to know computer science? That a computer scientist needs to know statistics? That both should be numerical analysts? Not that it would hurt. But the point is simply that there is an area in which a statistician with a good knowledge of computer science, or a computer scientist with a good knowledge of statistics, can make a valuable contribution. The remainder of this paper will briefly describe some of the specific activities I have in mind.

The design of software for statistical analysis of data is the most prominent area of statistical computations, and is probably what most people think of when the term is used. In a recent survey (1) Schucany, Minton and Shannon review thirty-seven statistical packages. They mention eighteen others not reviewed. There are others in neither list, and new ones being devised. Certainly development of such packages should require knowledge of both statistics and computer science.

Consider, for example, the problem, which some of these packages entail, of designing a "language" for specifying statistical processing operations. A knowledge of statistics is needed in determining the semantic content of the language: What operations are needed? Upon what sort of operands? But the already extensive knowledge of programming language design and implementation, which is part of computer science, cannot be ignored.

One might think that with all these packages available, no more statistical computations of this nature should be needed. As I look through all the available packages I begin to feel sympathy for the professor at my University who periodically proposes a moratorium on new course offerings. I have always been tempted to follow his proposal with one that the year be changed to 1938. Neither of these proposals stands much chance of being adopted, and as much as I would like to see a more consolidated and cooperative approach to the development of statistical software, I do not expect it. Perhaps the proliferation of statistical packages is evidence that no one has, so far, done the job well enough.

The history of statistical methodology seems to show that theory lags behind practice. A given analytic tool is put into use by people with data to analyze even though there is little formal support for this use. Later the theoretical work may prove that what is being done has some validity, and perhaps will delimit the conditions under which the technique is appropriate. By giving the user of statistics a powerful arithmetic enhancement, the computer may seem to have intensified the effect of this lag between practice and theory. As an example, techniques for reducing the dimensionality of multivariate problems, such as factor analysis, are in wide use even though their theoretical justification is not very firm. It may seem unfair to the theoretical statistician to give the immense power of computer arithmetic to the computer user without providing compensatory help for the theorist. However, the computer does offer some help in the development of the theory as well. One way this happens is by making it possible to evaluate expressions which at one time would have been intractable. Thus the range of numerically useful theoretical results is expanded.

A second aid to the development of theory is the use of Monte Carlo experiments. Current statistical literature offers many examples in which properties of statistics have been obtained or compared empirically through such experiments. Since these experiments will require large samples and hence a great quantity of computation it is necessary that a certain amount of computer programming sophistication be applied to them. On the other hand, for the same reason, statistical knowledge of sampling technique, variance reduction methods etc. will be very useful. Quite often neither of these aids is much in evidence. It would be perhaps a good idea to include this topic in courses on sampling.

Other help to theoretical work is available from the computer. Programming languages for the manipulation of symbols are in use and may be of some help to statisticians. General purpose languages with character manipulating capability, such as PL/1, are convenient for programming particular algebraic problems where the manipulations are simple and definite but where the great number of symbols to be operated upon makes it impossible to do the work by hand in any reasonable length of time. A similar application of the computer can be made to the many combinatorial problems which arise in statistics. Many of these problems relate to experimental designs. The computer has been used to count latin squares, count the number of occurrences of values of the determinant of zero-one matrices, to find the association matrices of PBIB's and in other such design related applications.(2,3) The computer has also been used to obtain symbolic relationships among the various symmetric functions encountered in moment calculations for sampling from finite populations.(4)

To illustrate some symbolic computing applications, the following figures show output from computer programs which have been applied to these kinds of problems.(5) The first shows the approximate mean and variance of a function

of a random variable obtained by the method of "statistical differentials". The program is written in the FORMAC language (6). The input is the function whose mean and variance is desired, the output symbolic expressions for the mean and variance. The second figure gives the density function of a transformed random variable. The input is the density function of the original random variable and the transformation to be made. The output is the density of the transformed variable. This program was also written in FORMAC. The third example shows the output from a PL/1 program which gives symbolic expressions for the expected mean squares in pure random sampling from balanced complete finite populations. The fourth example shows the output from a program which is being used to study incomplete block designs. The block-treatment structure is given as input. The program, rather perversely, attempts to find the association matrices for the design formed by interchanging treatments and blocks. The output gives the association matrices and the parameters (the p_{jk}^{i}'s) for the design.

APPROXIMATE MEAN AND VARIANCE OF A FUNCTION

$$FX = \sin (X(2)) X(3) X(5) X(4) + \cos (X(2)) X(1)$$

APPROXIMATE MEAN

$$UF = \sin (U2) U3\ U4\ U5 + \cos (U2) U1$$

APPROXIMATE VARIANCE

$$VF = \sin^2 (U2) U3^2 U4^2 V5 + \cos^2 (U2) V1 + (\cos (U2) U3\ U4\ U5 - \sin (U2) U1)^2 V2 + \sin^2 (U2) U4^2 V3\ U5^2 + \sin^2 (U2) U3^2 V4\ U5^2$$

Fig. 1 Mean and variance by statistical differentials.

DENSITY:

$FX = 1/6 (- X + 1) X$

$A = 0$

$B = 1$

TRANSFORMATION:

$Y = X^2$

DENSITY OF Y:

$G = 1/12 (- Y^{1/2} + 1)$

IN THE RANGE 0.00000E+00 < Y < 1.00000E+00

DENSITY:

$FX = 2/9 (X + 1)$

$A = -1$

$B = 2$

TRANSFORMATION:

$Y = X^2 - 3$

DENSITY OF Y:

$G = 1/9 (- (Y + 3)^{1/2} + 1) / (Y + 3)^{1/2} + 1/9 ((Y + 3)^{1/2} + 1) / (Y + 3)^{1/2}$

IN THE RANGE -3.00000E+00 < Y < -2.00000E+00

$G = 1/9 ((Y + 3)^{1/2} + 1) / (Y + 3)^{1/2}$

IN THE RANGE -2.00000E+00 < Y < 1.00000E+00

Fig. 2 Density of a transformed random variable.

FACTORS INPUT:
A.A(C).AB(D).B

$$EMS((A)) = BCD\ V_{(A)} + (CD-BCD/B^*)\ V_{(AB)} + (BD-BCD/C^*)\ V_{A(C)}$$
$$+ (D-BD/B^*-CD/C^*+BCD/C^*B^*)\ V_{A(CB)} + (C-CD/D^*)\ V_{AB(D)}$$
$$+ (1-C/C^*-D/D^*+CD/C^*D^*)\ V_{AB(CD)}$$

$$EMS((B)) = ACD\ V_{(B)} + (CD-ACD/A^*)\ V_{(AB)} + (D-CD/C^*)\ V_{A(CB)}$$
$$+ (C-CD/D^*)\ V_{AB(D)} + (1-C/C^*-D/D^*+CD/C^*D^*)\ V_{AB(CD)}$$

$$EMS((AB)) = CD\ V_{(AB)} + (D-CD/C^*)\ V_{A(CB)} + (C-CD/D^*)\ V_{AB(D)}$$
$$+ (1-C/C^*-D/D^*+CD/C^*D^*)\ V_{AB(CD)}$$

$$EMS(A(C)) = BD\ V_{A(C)} + (D-BD/B^*)\ V_{A(CB)} + (1-D/D^*)\ V_{AB(CD)}$$

$$EMS(A(CB)) = D\ V_{A(CB)} + (1-D/D^*)\ V_{AB(CD)}$$

$$EMS(AB(D)) = C\ V_{AB(D)} + (1-C/C^*)\ V_{AB(CD)}$$

Fig. 3 Symbolic expected mean squares.

ORIGINAL DESIGN PLAN 11.5

BLOCK	TREATMENTS
1	1 2 3
2	4 5 6
3	1 2 4
4	3 5 6
5	1 2 5
6	3 4 6
7	1 2 6
8	3 4 5
9	1 3 4
10	2 5 6
11	1 3 5
12	2 4 6
13	1 3 6
14	2 4 5
15	1 4 5
16	2 3 6
17	1 4 6
18	2 3 5
19	1 5 6
20	2 3 4

P MATRICES

P(0)

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

P(1)

0	1	0	0
1	0	0	0
0	0	0	1
0	0	1	0

P(2)

0	0	1	0
0	0	0	1
9	0	4	4
0	9	4	4

P(3)

0	0	0	1
0	0	1	0
0	9	4	4
9	0	4	4

Fig. 4 Parameters of the PBIB formed by exchanging blocks and treatments of a BIB.

Another application area which is very active and which is suitable for a combined attack using statistics and computer science is that of computer simulation. Computer simulation is accomplished by constructing a programmed moedl of some real life phenomenon or system. When such a computer model is constructed it is usually used for performing experiments, often random experiments. A great deal of statistical work needs to be done in the design and analysis of such experiments. The usual statistical methodology does not take into account the dynamic nature of simulation experiments. In a typical designed experiment there are several independent variables and a dependent variable. This is often not the case for simulation experiments, which often contain feedback loops, and which produce multiple time series as output. Work on the design and analysis of simulation experiments would seem to be an appropriate application of statistical and computer science interest in combination.

Most of the activities mentioned above emphasize use of the computer for statistical purposes. There is also a need for using statistics by the computer scientist. One aspect of this work is in the evaluation of performance of operating systems. Modern computer systems are capable of combining many tasks in a single "central processing unit". Scheduling and controlling the variety of things which are going on simultaneously or intermittently is the task of a program called the operating system. How will an operating system work given a mixture of jobs from a certain population? What parameters are most important to performance? A statistician should be able to offer help in investigating such questions.

A perhaps simpler problem relates to the performance of algorithms. People who study computational complexity may be able to show the existence or non-existence of certain algorithms or classes of algorithms. They may also be able to prove that they run in a certain number of operations. For example,

$O(n^3)$ operations to invert a matrix, O(nlogn) to sort a list. These numbers of operations are usually based upon worst case analyses, but there may be times when it would be useful to assume a distribution for the problem data, and find the expected number of operations, the variance, the probability of requiring more than N operations, and the like.

A problem that every programmer has is "proving" that his program works. If you have used the computer to a good extent, really pushing the software at times, you will find, once in a while, a bug in a program which has been in use for a long time and is supposed to be completely debugged. A computer program is a tree structure. Depending upon input, it will take one branch one time, some other another time. If the program has n branch points we need 2^n sets of data to try them all, but programmers get tired. Furthermore it may not be evident which 2^n sets should be used. Does statistics offer any help for this problem?

The final area which I want to mention is that of "artificial intelligence". I have some hesitancy about using this term in mixed company because it seems to stir up an unwarranted amount of contention. I recently received an account of a faculty meeting in which a proposed course in artificial intelligence was discussed. Almost immediately the existence of such a thing was challenged. A long controversy ensued which appeared to demonstrate that the existence of <u>any</u> kind of intelligence was extremely doubtful. The term artificial intelligence, used informally embraces a range of interconnected topics including pattern recognition, machine learning, adaptive systems, theorem proving, natural language processing, heuristic programming, etc. Many of these activities involve the solution of problems which are statistical in nature: Estimation, discrimination, prediction and the like. Whatever your view of the potentiality of machines to do thoughtful things, these are

interesting and important problems. One reason for this is that the analysis of problem solving activity which they involve may give additional insight into how problems may be solved, by humans or machines. This is particularly important for statisticians, because many of these problems involve simulation of inductive reasoning. Another reason for the importance of these problems is that they interact with other uses of the computer. For example consider the use of symbolic computations. Such computations involve the manipulations of expressions using the rules of algebra and produce other expressions. The reason for using the computer to do this is that the expressions involve many terms, many operations. The final product is likely to be a long string of symbols, but will it be useful? Can the computer be programmed to formulate these expressions in a way which will be helpful to the investigator who caused them to be generated? There are of course many other possible applications of the solutions of artificial intelligence problems. Many of them fall in this same area of aiding communication between humans and the computer.

I have tried to mention several areas where knowledge of statistics and computer science can be combined with beneficial result. I do not ask that all statisticians be computer scientists, nor that all computer scientists be statisticians. If I were going to ask for anything, I guess it would be that they treat each other with mutual respect.

References

1. W. R. Schucany, Paul D. Minton, and B. Stanley Shannon, Jr., A Survey of Statistical Packages, Computing Surveys, Vol. 4, No. 2, June 1972.

2. Mark B. Wells, Elements of Combinatorial Computing, Pergamon Press, 1971.

3. T. J. Mitchell, An Algorithm for the Construction of "D-Optimal" Experimental Designs, Applied to First-Order Models, Oak Ridge National Laboratories ORNL-4777, May 1972.

4. E. J. Carney, Polykays and Ordered Partitions--Some Computing Problems, in Symmetric Functions in Statistics, Derrick S. Tracy, Ed., University of Windsor, 1972.

5. E. J. Carney, Symbolic Computations and Statistical Problems, Computer Science and Experimental Statistics Technical Report No. 72-107, University of Rhode Island, September 1972.

6. R. Tobey et. al., PL/1-FORMAC Interpreter User's Reference Manual, IBM Contributed Program Library 360D-03.3.004, Hawthorne, New York, 1967.