Statistical Science
1986, Vol. 1, No. 2, 259–275

Computing Environments for Data Analysis

Ronald A. Thisted

Abstract. A computing environment is a collection of hardware and software tools that are integrated so as to work well with one another and designed to make a selected class of computations easy and natural to carry out. Experience with interactive statistical programs such as GLIM, Snap, and Minitab suggests that experienced data analysts develop distinctive computing styles when approaching data that integrate some of the tools these programs provide into more general strategies for data analysis. Computing environments specifically designed for data analysis make these strategies easier and more natural to describe and to implement, and make them accessible to data analysts with considerably less experience.

Key words and phrases: GLIM, interactive statistical software, integrated computing environments, Minitab, statistical computing, statistical strategy, software.

1. INTRODUCTION

In the 1940's, the computations required for data analysis were done using mechanical calculators, pencils, and stacks of graph paper. In the 1950's, these same computations—and others hitherto impossible—were accomplished through specially written programs for large computers. In the 1960s, packages of statistical programs for large mainframe computers became available, making even newer and more complex things possible, and establishing a path to statistical computation that could be followed even by nonprogrammers and nonstatisticians. In the 1970's, interactive statistical programs became widely available, and for the first time it became possible using the computer to duplicate the high degree of interaction with data characteristic of statistical computing in the 1940's. In each of these decades, major changes have occurred in the way computations associated with data analysis have been carried out. Each of these changes has made it possible to do more thorough and more penetrating analyses while, at the same time, making it possible for a greater number of scientists to use statistical methods profitably. Each, too, has been a logical outgrowth of its predecessor. The decade of the eighties will be no different in this respect.

In the 1980's yet one more forward step is being taken—creation of *computing environments*. We can view the computer terminals, the statistical software packages, the programming languages, the editors, the operating systems, and the output devices and displays as separate "tools" used to carry out portions of a data analysis. A computing environment is an integrated collection of such tools. By "integrated" we mean that the tools are designed to work well together so as to make a selected class of computations, namely the ones we employ in data analysis, easy and natural to carry out.

The sorts of things that *should* be easy and natural will, of course, depend on the nature of the data analysis—the kind of data being examined, the kinds of models being entertained, the dimensionality of the data set. A good computing environment will make it easy to access those tools which are appropriate in a given context, and to shift from one to another as more is learned in the process of data analysis itself. It will help to consider one or two familiar interactive programs for data analysis and to examine how they could be integrated into a larger collection of tools to make data analysis using them easier, more natural, and more productive. The two programs we shall use as examples in the sequel are GLIM and Minitab, the former because it is an extremely flexible and powerful program capable of examining quite general models including both continuous and discrete data, the latter because it is extremely easy to learn and to use, even for those with little training or experience in statistics and computing. (Unfortunately, GLIM is not easy and Minitab is not powerful; a good computing environment is both.) Some preliminary comments about GLIM will set the stage for further discussion.

The acronym "GLIM" stands for generalized linear interactive modeling (Nelder and Wedderburn, 1972). This is as good a place as any to note that by "data analysis" we really mean *modeling* in the broad sense,

Ronald A. Thisted is Associate Professor, Department of Statistics, University of Chicago, 5734 University Avenue, Chicago, Illinois 60637.

incorporating such aspects as tentative model identification, diagnostic checking, transformation, and data cleaning, as well as the usual inferential procedures such as estimation of parameters and their standard errors associated with a particular model. It is the interactive modeling aspects of GLIM that will be of interest to us in our discussion of computing environments.

At a conference on generalized linear models held at the University of Texas at Austin in June 1984, Carl Morris raised the question in his keynote address, "Is GLIM a package or a concept?" The answer is that it is both. GLIM's success as a package derives in part from the fact that it enables us to adopt a particularly helpful *approach* to data analysis. This same assertion cannot be made about most of the statistical program packages in wide use today, particularly the batch statistical computing programs.

There are two aspects of the Nelder/Wedderburn approach to modeling: it provides a framework which unifies exponential family regression, and it provides software which makes it possible as a practical matter to adopt the framework. This framework for analysis is well described in McCullagh and Nelder (1983). Thus, we have a powerful and general approach to modeling a wide variety of responses depending in quite general ways on covariates *and a way to put that unified theory into practice.*

The GLIM "theory" is really a paradigm. Namely, analysis proceeds by thinking separately about three aspects of the model: the random component Y, which has mean value μ and variance function $V(\mu)$, the systematic component $\eta = \sum \beta_j x_j$ which describes the dependence on the covariates X, and a link function g which relates the mean of the random component Y to the systematic component η by $\eta = g(\mu)$. It is this separability of components that makes GLIM a powerful general tool. In effect, this separability helps us to structure our approach to data analysis by allowing us first to break the problem into small, distinct "chunks," and then to focus on each of these chunks in turn. This structured approach to data analysis is close in spirit and practice to the modern "top-down" approach to programming.

Might a batch program which fits such models do just as well as GLIM for our purposes? The answer must be, "No!" GLIM is essentially interactive, and that is an intrinsic aspect of its usefulness and appeal. Interactive programs are typically built around a collection of atomic commands, each of which performs some small unit of work. Because the sequence in which commands will be entered in any session is not known in advance (even to the user!), each command must represent a small step in the analysis (otherwise, there would be no need for interaction). Exploratory data analysis must by its very nature proceed in just such small steps. Indeed, modeling is an inherently interactive process, requiring as it does constant checking of tentative models and fitting of plausible alternatives. Moreover, what is learned up to any given point in the analysis can and should affect the course of what follows. One model for the process of modeling is that of traversing a *search tree*, where at each node we must decide among several choices for such things as variable selection, outlier identification, and choice of link function. It seems clear, then, that if GLIM is to be incorporated into a more general computing environment, the latter will necessarily be interactive, and perhaps will be more so (in some sense) than GLIM currently can be.

The GLIM program, then, is the *tool* that we need to implement a new and powerful *strategy* for data analysis. In general, this is precisely the role that software should play in statistics. Past efforts in statistical computing have been directed toward implementing statistical methods rather than data analytic strategies. We are now at the point in our understanding both of data analysis and of computing systems that we can and should think seriously about the latter. Our theme, then, will be that good computing environments consist of software tools, integrated with appropriate hardware, which implement strategies for data analysis.

2. WHAT IS A COMPUTING ENVIRONMENT?

We have loosely defined the idea of a computing environment in terms of three criteria: a collection of *integrated* hardware and software *tools* designed to make a selected class of related computations *easier or more natural* to carry out. The fundamental tools of which such environments are built need not be—often they should not be—highly complex. Indeed, gluing together relatively primitive components is often all that is needed to produce a quantum leap in the utility of the result. Some examples of computing environments may help to make this point more clearly.

Example

Lotus 1-2-3 is a popular program for business users of personal computers such as the IBM PC. Some argue that it is single-handedly responsible for the great acceptance of personal computers in the managerial office. Lotus 1-2-3 might be termed an environment for business users. It integrates three tools, each of which is fairly primitive in absolute terms: a data base manager, an electronic spreadsheet, and a business graphics program. ("Business graphics" programs typically allow one to draw pie charts and bar charts, and to adorn them with color, labels, and fancy grids; such capabilities are rarely useful in data analysis.)

Each of these is a rudimentary tool for business analysis or communication, and it is common to use each of them in the course, say, of examining options to buy or to sell a property under a range of assumptions. The forward step that Lotus 1-2-3 represents is that its authors recognized that the output from any one of these three activities might well be used as input to another at a subsequent stage of analysis. So what Lotus 1-2-3 does is to make it easy and natural for the business manager to combine different (complementary) analyses based on different data sets and on alternative assumptions. What it has done is to eliminate the time- and knowledge-consuming steps required to move data from one application to another, thus making it possible for the business person to spend more time doing analysis and less time learning about computing systems. The leap in productivity achieved by linking these three primitive activities was enormous, partly because of the time saved, but partly because the analyst no longer needed to divert his or her attention from the questions of primary interest to deal with unrelated, but utterly essential, aspects of operating systems and data formats and the like.

Example

UNIX, together with the *Programmer's Workbench*, is a computing environment for software development. These tools make it easy and natural to track different versions of a program under development, to track differences between versions, to insure that the most recent version of a module will always be used, to document programs being developed and to keep the documentation up to date, and to communicate with others working on the same project. UNIX enhances programmer productivity by relieving the software development team of the most tedious—and essential—tasks in the process. In effect, the programming environment provides much of the glue needed to hold the project together.

Example

The *S* statistical system is an example of one approach to designing an environment for data analysis (Becker and Chambers, 1984a, 1985). *S* is an open-ended program which allows users to link together primitive constructions in order to build special purpose tools (which can then be used themselves as building blocks for further extensions). The system is specifically designed for interactive exploratory work; details of the system design are given in Becker and Chambers (1984b).

The "objects" which we compute with (or construct) in data analysis are not easily expressed as rectangular arrays. For example, the output from a regression analysis typically will include a vector of fitted values, a vector of residuals, a vector of estimated regression coefficients, a matrix of variances and covariances of the latter, and a set of observation numbers indicating those observations in the data set worth further examination. This output collection should be treated *logically* as a single entity with important constituent parts. The classical $n \times p$ rectangular array can certainly be used as a *container* for these constituents, but only at the expense of having the user of the array do some bookkeeping on the side to keep track of what is in the array, and where. Of course, such bookkeeping work has nothing whatsoever to do with data analysis.

S was designed to allow completely general hierarchical data structures. What is more, the user of *S* need not be aware of what individual structures look like in order to use the system. This serves two important purposes, each of which serves to increase human productivity. First, for the builder of new tools, it becomes easy to incorporate natural data structures without having to construct an appropriate representation of the data built from standard data structures. Since the data structures can be extended by users, the user can make his or her data structures reflect the way he or she thinks about the problem instead of being constrained to think about the problem solely in terms of those data structures which may be available. Second, for those using *S* for data analysis, the details of data management are hidden, so that there is less technical material that needs to be learned as a prerequisite to using the program, and more of the time spent at the computer can be devoted to productive activities.

Example

DART (Donoho, 1983) is an interactive statistical computing language that extends the *S* language, adding special features such as the ability to debug functions, and introducing notations for referring to and making assignments with general tree-like structures. DART is optimized for programming power rather than natural interaction during data analysis; it is designed to provide tools for expert data analysis with which programs for particular data analyses can be built. As such it is a promising environment for *statistical programming.*

Becker and Chambers note that, "the effectiveness of the *human* is the most important criterion for design of a computer system" (1984b, p. 488). So the central question is this: How should statistical software be designed so as to perform these basic tasks of integrating the fundamental tools we use and of improving our effectiveness as statisticians by taking on essential support tasks? To answer this question, we must examine the sorts of tools that statisticians use, and then we need to look at what sorts of nondata

analytic work we often must do in the course of data analysis.

3. WHAT ARE THE TOOLS OF DATA ANALYSIS?

Whenever we sit down to examine a data set from a new problem, we make use of many features of the computing systems available to us. These features include computer terminals, telecommunications, computer hardware, operating systems, workstations, programming languages, editors, statistical packages, special purpose programs, printers, and plotters. We shall conduct a brief examination of some of these basic tools and of how they are used in data analysis.

For most statisticians, the computer hardware, the telecommunications links, and the operating systems represent merely the substrate on which data analysis is conducted. For most, these aspects of their total computing environment are given and relatively unchangeable. What is more, only rarely have these components been designed with data analysis specifically in mind. Rather, they represent general purpose structures for general purpose computation. In the past 5 years, efforts have been directed toward developing computer hardware configurations, and operating systems for those configurations, for the specific purpose of doing data analysis. A good recent example is the ORION-1 workstation developed by Jerome Friedman, Werner Stuetzle, and their colleagues at the Stanford Linear Accelerator Center in the early 1980's (Friedman and Stuetzle, 1983).

In general, workstations are single-user computers integrated into a hardware system designed for a special purpose. In addition to the ORION-1 data analysis workstation, other examples include the SUN and Apollo graphics workstations, and the Symbolics Lisp machine, designed for developing software for symbolic computation. The key advantages which such workstations provide are not *directly* applicable to data analysis, although they have great potential as the medium in which effective data analysis environments may be realized. The power inherent in workstations such as the SUN or Apollo lies in synergies achieved by combining three categories: high bandwidth communication with other resources on a computing network (including other workstations), the ability to use a wide variety of applications software, and a graphical interface to those applications which supports multicontext use (Joy and Gage, 1985). LISP machines such as the Symbolics 3600 have superb *programming* environments, some features of which can be adapted to data analysis. Here, too, the potential for data analysis lies in the window/menu systems which allow rapid and natural contex switching between multiple concurrent tasks. As of now, there is no generally available software for data analysis which makes intrinsic use of any of these distinguishing capabilities of workstations.

By developing appropriate software (and perhaps adjunct hardware), a general purpose workstation can serve as the basic tool for a comprehensive special purpose computing environment, such as one designed for interactive data analysis. Whatever the software looks like, to be effective it must coexist well, if not synergistically, with the underlying hardware. Although the remainder of this paper will focus primarily on software aspects (since the goal of implementing statistical strategies seems more amenable to software than to hardware approaches), we shall also raise the matter of suitable hardware and its integration into a unified system where it is appropriate to do so. A detailed and informative series of papers by McDonald and Pederson (1985a, 1985b) discusses hardware considerations in constructing environments for data analysis.

The data analyst sometimes has a choice of computer terminals to use, and there is a wide array of choices available. There are paper copy terminals, ranging from inexpensive dot matrix devices to letter quality impact terminals. There are display terminals of every stripe, from the "glass teletypes" to intelligent editing terminals; from high resolution graphics devices to full color instruments. These display terminals often have facilities for producing permanent copies of screen displays. It is interesting to note, however, that only in the past year or two have commercial statistical software products begun to have the capability of using any features at all beyond those of the "glass teletype."

On the software side, much data analysis is currently done using statistical packages such as GLIM, ISP, Minitab, Snap, SCSS, IDA, *S*, or any of a host of other interactive programs. These programs each have their strengths, although none is entirely adequate for every data analysis. It is often desirable to shift from one package to another in the course of an analysis, although it is rarely feasible to do so in practice. Another frequent occurrence is that a separate computer program must be written, either to solve a formatting problem, or to do a side calculation, or to carry out more complicated or special purpose analyses than the standard packages provide. For the most part, it is difficult or impossible to move from one of these tools to another.

We also find ourselves using features of the operating system and other, nonstatistical software to perform some of our basic work. In particular, system sorting facilities and text editors are essential for such routine tasks as entering and screening a data set. Again, these tools provided by the operating system are rarely accessible at times when they would be most

useful, say, while in the middle of examining a data set using Minitab.

In short, then, what we often have at our disposal is an *unintegrated* collection of highly useful tools. Our next step is to examine how statisticians use these tools in practice, in order to suggest fruitful ways of integrating them. It is important to note that a suitable integrated system will provide the capabilities of today's tools without necessarily providing them in a form recognizably derivative from today's standard computing packages.

4. WHAT DO DATA ANALYSTS DO?

A prerequisite to constructing a computing environment is to understand what sorts of things should be easy to accomplish, and what are natural ways to think of data analysis problems. This way of thinking about the problem represents a behavioral approach to software design. Rather than focus on the particular *computations* that we want the package to do—that is, the particular statistical analyses we want to have implemented—we focus instead on the program's *interactions* with us while we do our work. The emphasis is on how we behave when we work with data, and on how a program can be designed to mesh comfortably with our own behavior. To some extent, if this plan were successfully carried out it would require that a different program be written for each data analyst. That is not a bad goal, nor is it as far fetched as it may at first sound. What it suggests is that a good environment will fit many people well, and will have sufficient flexibility that it can be tailored to accommodate differences in computing style. Thus, in designing a system, we must consider both aspects: it must be generally useful and must be "customizable." We first address the former.

The best way to find out what data analysts do is to watch them at work. Since many people, data analysts included, do not enjoy having someone look over their shoulders, one place to start is with some self-observation. (We can look in on some other data analysts a little later, and we shall mention some technical devices for doing so in Section 6.) In this introspective phase, the question we should have in the back of our minds is this, "What strategies do I adopt when I do data analysis, and what must I now go through in order to implement those strategies?

Whether we observe ourselves or others, there is a general approach that we can take. Watch how a user interacts with one or more interactive statistical programs, preferably over the course of an entire session, from first look at the data to last. Note particularly aspects of what might be termed a "computing style": the ways in which features of the package and of the operating system are combined with one another. Note the frustrations encountered in using these systems. Note the *ad hoc* devices (that is, shortcuts) that the user invents in order to get the job done. These shortcuts usually represent aspects of the problem that are so basic—or so useful—that they have merited some intellectual effort diverted from the main task at hand in order to make them easier to get done. What we are watching for in all of this are peripheral, but essential, tasks from which the data analyst might be spared, making it possible to concentrate more completely on the tasks of data analysis.

Let's work through an example, based on introspection but not too different, I hope, from scenarios recognizable to most data analysts. I often use Minitab (Ryan et al., 1976) to take a first look at data. I log in to my local computer through a display terminal, preferably one which will communicate at high data rates (4800 baud or faster). I also sit down with pencil and a pad of paper. If the data set is a small one, I enter it directly into the Minitab worksheet. If I find that I have mistyped an entry, or have left one out, I usually end up retyping a substantial fraction of the data set, so I try to be very careful when I enter the data. If, on the other hand, the data set is moderately large, I first create a data file using my favorite editor, and then I enter Minitab. If I then find errors, I leave Minitab, go back to the editor and make corrections, then I return to Minitab.

In the course of the analysis, I often divide the observations into subsets, or I remove one or two points provisionally. I also will frequently examine the effect of one or more transformations on some of the variables in the data set. At any given point, I have several alternative versions of my data with which I am working: the original (painstakingly typed), the version with $\ln(y)$ instead of y, the version which omits Alaska and Hawaii, the version which includes only those individuals with a family history of allergy. To construct these derivative data sets requires considerable use of Minitab's PICK and CHOOSE commands, and before long, columns containing portions of these data subsets are scattered throughout the worksheet. With each of these derived data sets I perform a few statistical computations (such as fitting a least squares line), and I draw a few plots (such as a scatterplot or a residual plot).

A very small number of these analyses are of sufficient interest to keep for reference. What I must do to obtain a paper copy of those few depends on the particular device I am using as a terminal. If I am using a personal computer with a terminal-emulation program, then I can press a key which will start saving a transcript of the terminal session on a floppy disk, the contents of which I can later print out. If I am using a plain vanilla display terminal (that is, one with no hard copy facility or page memory), then I

must have Minitab direct all of its output to a file, repeat the analysis, and then leave Minitab. At that point, I can direct the file to a printer which, for me, is located a 10-min walk away. Alternatively, I can log off the computer, then log in again using a slow hard copy terminal on which the saved file can be typed out while I wait (and wait). Finally, if I am working at a terminal which has a printer of its own attached to it, I simply press a key to obtain a hard copy of each potentially useful display.

By the end of this process, the pad of paper next to me has been filled. What is on the paper? First, it contains notes about intermediate results, created variables and, what is most important, a *map* through them. Without this map, I would lose my way in the plethora of newly created variables and data subsets. The paper, therefore, keeps track of which variables have been placed in which columns, and some notations of how columns are related to one another. Second, since the very notion of modeling implies repeated comparison, I have notes which keep track of my current best candidates and of aspects of them worth noting, such as values of the deviance, or the residual standard deviation, or possible outliers, or possible transformations. These are the notes I need in order to compare one model to another, both in terms of goodness of fit and in terms of qualitative differences. Third, the paper contains a "stack" of possible next steps, and some notations as to which of these are most promising. At each stage in the analysis it is possible—even likely—that the results so far will suggest two or more paths to follow. Only one of these paths can be explored at a time, so I make notations about things to come back to later. A useful way of describing what is done in interactive modeling is to say that a *search tree* is being constructed. The paper notepad keeps track of our current position and of our plans for further exploration in this tree. Finally, if I have access to immediate hard copies of what appears on my screen, my notebook will contain annotated copies of useful plots and other displays such as tables of regression coefficients.

The notebook, then, contains a fairly detailed record of the process and progress of the actual *analysis* (as opposed to computation). It records the information necessary to reconstruct the context within which each of the computations was performed. Part of this record is valuable after the computer session is complete, for example notes comparing features of alternative models, and annotated output from Minitab. Most of the record is ephemeral, reflecting bookkeeping chores that were needed to keep from losing track of where we were, what we had done, and where we had put things. These latter portions of the record all required substantial effort and some planning; performing these tasks diverted attention and energy from the modeling process itself. These tasks can be minimized or eliminated entirely by an appropriately designed computing system. In the next section we examine some ways in which a computing environment for data analysis could do so.

5. DESIGNING A DATA ANALYSIS ENVIRONMENT: SOME EXAMPLES

Now that we have examined a typical computer data analysis session, we can identify some rough spots in the process, and imagine how a better computing system might alleviate or eliminate those difficulties. First we consider some "brute force" methods of improving the computing environment in the context which is most widely available to statisticians today, namely the timesharing computer system. We then proceed to examine how more carefully tailored environments based on more recent (but more expensive and less available) technology can improve matters still further.

5.1 Interactive Timesharing Systems

The very first thing we do is to decide whether to use a fast display terminal without hard copy or a slow terminal which provides a permanent log of the session. We have noted that speed is essential. If we have to wait two minutes for Minitab to produce each plot, we will do less plotting and we must twiddle thumbs (or the mental equivalent) while waiting for each picture to be produced. At the same time, permanent copies of portions of the computed output are also essential. Right now, we must choose which aspect has higher priority, and then make do with respect to the other.

Most statisticians choose display speed over hard copy and with good reason. The matter of computer speed is not simply a convenience, it affects the quality of the data analysis. Even short periods of time spent waiting for output tend to divide the total effort devoted to the actual analysis into disjoint fragments, and fragmented time is rarely well spent. An hour of concentrated effort is usually more productive than that same hour spread over twelve 5-min intervals during the course of a day. In each working interval, there is a period of gearing up at the beginning and tidying up at the end, both mentally and physically. Long waits for portions of the output encourage time fragmentation. When the plot is finally done, whatever other activity that has been done in the meantime must be interrupted and the data analysis process resumed.

Permanent copies of some subresults are necessary, however, since we need to compare results such as shapes of regression curves, or magnitudes of coefficients, or sizes of residual standard deviations, across

two or more models. A common expedient is to copy down important numerical results onto a scratch pad. This solution also causes time fragmentation, but since it occurs relatively less frequently than would long waits at a hard copy terminal, the trade off is usually deemed acceptable. The most difficult analyses to do are those in which these two considerations are roughly in balance.

The conflicting demands of speed and reference copy availability can be met by a computing environment that addresses both issues simultaneously. A colleague of mine, David Draper, deals with the problem by logging in to the mainframe computer system simultaneously on a fast display terminal *and* a slow hard copy terminal. He then switches back and forth between the two, depending on his particular needs at the moment. Doing this requires some cooperation from the computer's operating system, such as the ATTACH command provided by the TOPS-20 operating system, which allows one to switch a computer session from one terminal to another.

Many display terminals now have an output port built into them through which the contents of the screen may be dumped to a printer (or even to a hard copy terminal). With such a configuration, it makes it possible to dump fragments of text or plots directly to a hard copy whenever desired; the only limitation is that the entire image to be preserved must fit on a single screen. This is sometimes a limitation when "printer plots" are produced by programs such as Minitab. Graphics terminals have for some time had hard copy units attached to them with the capability of taking a snapshot of the display screen's current contents. Their major drawback has typically been their cost, both the original hardware and the cost per copy. This difficulty is rapidly vanishing.

5.2 Windows

The two methods described above make use of existing technologies, albeit using brute force. A third possibility for addressing this problem—not now available in any generally available system for data analysis—is not to have two *physical* terminals at all, but rather, for a single device to emulate a pair of terminals that behave in an appropriately integrated way. Today's workstations make this possible. In effect, we use a single display the screen of which can be divided into two or more regions (called "*windows*"). Separate tasks are conducted in separate windows, and one switches tasks by moving the cursor from one window to another. For example, when the cursor is in the first window, it is as if we have logged in and are using Minitab in the usual fashion. What appears in the first window is just what would appear on the terminal screen during an ordinary Minitab session. In the second window a "notebook" program in running; when the cursor is in this window, we are able to enter annotations, and to copy portions of what appears in the first window. Moreover, the contents of the second window are continuously and automatically saved in a file for later reference during the computing session. We can think of a three-ring binder as being a metaphor for the second window. Whenever we desire, we can 1) add a leaf to the notebook by copying parts of the first window to the second, 2) add a leaf by typing annotations or notes directly in the second window, 3) observe the most recently created leaf by staring at the second window, and 4) examine previously created pages by leafing through the notebook. These entries in the notebook would optionally be accompanied with an automatic time stamp.

Having a "fast" notebook right next to the fast display used for the analysis makes the entire process easier and more natural. Indeed, the plain vanilla approach to using Minitab (with just Minitab running on a single terminal) uses the computer to do only *part* of what we used to do by hand; the rest we still had to do by hand. The proposal in the previous paragraph integrates much more of the process. Indeed, a fast virtual notebook addresses another matter easily and naturally—the fact that much of what we write down on our legal pads is needed only during the analysis session, and is of little use after. With the virtual notebook, we can at any time discard those pages which are no longer of direct use and, at the end of the process, we are left with an archival document containing the important elements of lasting value from the analysis.

To implement such an approach requires appropriate hardware, software, and operating systems support. The amount of information that can be displayed on a single screen varies widely from one brand of terminal to the next. Since the terminal must at times display substantial portions of two windows at the same time, the amount of displayable information per window is cut at least in half, so the terminal will have to have an adequate information display density. Workstation displays typically meet this requirement; high resolution timesharing terminals which do so are only now becoming available. Under some circumstances it may be adequate to display completely only one window at a time, with portions of other windows "covered" by the active window.

A terminal with 24 lines of 72 characters is just barely adequate for many purposes, and may be inadequate for all but the most rudimentary graphics applications. If this area is cut in half, either horizontally or vertically, the amount of displayable information in each window becomes too small to be useful. Fortunately, screen density is increasing quickly, and it is relatively easy to find excellent inexpensive

display terminals with 30 or more lines vertically and 130 positions horizontally. Other terminals (and computers emulating terminals) have high resolution bit map displays which are more than equal to the task. The terminal, too, must provide for separate addressing and access to different portions of the screen, so that when activity takes place in one window, the other window need not be disturbed. (Without this feature, the entire screen would need to be redisplayed to deal with any change, raising the specter of having to wait uncomfortably while the screen is reconstructed.) Similarly, the operating system would have to provide support for multiple tasks, since, in effect, each window is controlled by a separate computer program, although these programs are linked to one another. The operating system is the most desirable level at which this linking should take place. Finally, the statistical analysis program itself should be designed with the understanding that it will be used in a multiple-window environment of the type described, so that it can provide more useful output, and so that the transition from one window to the other can proceed smoothly and naturally.

The notion of windowing is more generally useful in the design of data analysis software than simply providing an electronic replacement for the notebook on the side. Whenever there is a task to be done that is not comfortably done within the statistical program itself, or is more logically done using a separate program, that second computation can be performed in a separate window. Windows need not be simultaneously visible. The windows visible on the screen at any given time will be the ones corresponding to the tasks to which the analyst is currently devoting his attention; all of the other windows will be hidden, available on a "need to see" basis. In subsequent sections we shall suggest useful features of a data analysis environment, many of which can be implemented naturally using multiple windows.

5.3 Transcripts and Journals

One of the advantages of the hard copy terminal is that a complete log of the entire analysis session is available at all times, typically on a continuous sheet of fanfold paper spewed from the top of the terminal. The window metaphor can be brought to bear here, too. Think of the screen of the display terminal as a window being passed from top to bottom over the stream of fanfold paper; the paper hasn't vanished, but only the last 24 lines of its contents are visible. A display terminal with memory of its own can save the entire transcript and can allow the user to move the window "backward" so that it is once again over earlier portions of the terminal session. Many display terminals can be purchased today which have 16K bytes of memory and automatic scrolling of this sort built in. The Apple Macintosh computer running MacTerminal terminal emulation software can save as much of the terminal session as there is free space on the disk, which can amount to more than 200K bytes, all of which is instantly accessible at any time during the session. The same result can be achieved using the EMACS editor running under UNIX by initiating a new shell within an EMACS buffer.

Having such a continuously accessible *transcript* of the analysis session obviates much of the need for transcribing relevant portions of the analysis for later reference. An extra advantage of the transcript is that the portions of the analysis which are relevant often become apparent only later. The transcript feature, then, reduces the need for omniscience on the part of the investigator. This is actually an important aspect of a computing environment. While the experienced data analyst will have developed a feel for those aspects of the analysis to date which are likely to be useful later on in the process, the novice can rely on the transcript to supplement his maturing judgment. Thus, a good computing environment will facilitate both learning and doing.

The transcript contains both the commands to the statistical package and the resulting output. Since it may take only a few commands to generate several screen's worth of output, it is difficult to review the analysis as a whole (or even the computational part of the analysis) simply by reviewing the transcript. A separate, but closely related device makes this task much easier. A *journal* consists of a transcript of the *commands only* sent to the statistical package, together with automatic numbering. This device operates within the statistical program in much the same way that the *history* feature operates at the operating system (shell) level in Berkeley UNIX. The journal is continuously updated, and always available, but need not be always visible. When it is desired to review a portion of the analysis, the journal window is activated and then the entire set of commands is available for inspection.

Mere inspection of the commands may not be enough. If the session is very long or the terminal used has limited memory, it may not be possible to have the entire transcript available. In this case, it would be helpful to be able to "play back" portions of the session selected from the journal. The portion of the session to be replayed can be selected with a pointing device such as a mouse or trackball, or by explicit reference to command numbers. Indeed, the ability to replay a set of commands given earlier is a useful supplement to a macro facility, particularly since we often realize that we want to perform the same set of operations repeatedly *after* we have found them to be useful. (Once again, the experienced analyst may use a macro, recognizing in advance that a particular

sequence of commands will be repeated, while the less experienced user will be able to use the alternative facility of the journal window to achieve the same result.)

The contents of the journal window can, of course, be saved permanently as a text file which can then be annotated after the fact, or which can be "replayed" in order to reconstruct an analysis.

5.4 An Example

Minitab and other statistical programs could be extended to provide some of the facilities of a journal file, even when used on standard terminal equipment, by adding automatic command sequence numbers and two commands, SHOW and REDO. Thus, for instance, after entering 92 commands in a session, the user could type

[93]—show 25–30

which would then display

[25]—regress column c1 on 2 carriers: c18, c33 (c20, c21)
[26]—plot c20 vs c21
[27]—let c22 = c1 − c21 compute raw residuals
[28]—let c23 = c22 * c22 squared residuals
[29]—plot c22 vs c21 raw residual plot
[30]—plot c23 vs c21 heteroscedasticity?

To get the same analysis for the logarithm of the dependent variable, one could then simply enter the commands

[94]—let c1 = ln(c1)
[95]—redo 25–30

There are several alternative ways of implementing the same function, some of which do not involve modifying Minitab itself. One approach is to modify the operating system in such a way that history facilities are available within processes as well as at the operating system level. A second approach is to utilize "mark, cut, and paste" facilities provided by the operating system (as is generally available in Macintosh software). A third approach is to introduce an intermediate software layer which implements the necessary recording, selecting, and "pasting" features; an example is through EMACS shell windows. In the latter case, Minitab would be executed within one of several buffers (or windows) all under the control of a screen-oriented text editor.

5.5 Multiple Worksheets

Earlier we likened the process of data analysis to that of constructing and then traversing a search tree. Due to the sequential nature of most analysis software, most statisticians adopt a style of analysis that follows one branch of the search tree until it becomes apparent that further progress in that direction is unlikely. At that point the analyst backtracks to an earlier part of the tree and proceeds from there. Since the analysis progresses at any stage by modifying or adding to the current data set—say, by omitting points temporarily, or by introducing a transformation of the underlying variables—in order to return to a previous point in the analysis the effect of these modifications must be undone. Unfortunately, most current software packages make it difficult to "undo" work selectively. Rather, one must either save copies of the current session at all of the critical junctures (of course, keeping track of where each of these copies is saved, together with all of the necessary bookkeeping information needed to use each of them), or one must reconstruct the previous state of things by executing the inverse transformations. Cowley and Whiting (1985) and Gale (1985) discuss investigations on these topics currently under way at Pacific Northwest Laboratories and at AT&T Bell Laboratories, respectively.

In Minitab, the data set is thought of as a "worksheet" similar to what one would lay out on paper, were the analysis to be done by hand. A more general concept is that of a *workspace*, which can be thought of as a collection of linked worksheets. At any given time, a single worksheet is the *active worksheet*; the others are inactive. Each worksheet in the workspace can be assigned a name to which it can be referred. The key to making multiple worksheets effective is the linking structure, which must be automatically taken care of by the computing system.

One plan for organizing a workspace is to reflect the structure of the search tree being traversed. This means that the workspace itself would consist of a hierarchical structure of worksheets. Each node in this structure would consist of a worksheet derived by a series of transformations from its parent node (that is, parent worksheet), together with a *node history* containing the journal entries by which the transformation was accomplished (for reference purposes), and a "memo" in which the analyst could record notes about the direction being taken in the analysis up to this point. The "history" attached to the node would consist only of those commands which actually modified the data set; display and analysis commands would be omitted. At any time, the analyst would be able to visit other nodes to examine the worksheet there, to recall that node's history, to display data, or to continue the analysis from that node. In addition, at any time the analyst can create a new node, consisting of a new copy of the current worksheet, an empty history window, and a blank memo window. Such a structure would allow the statistician to undertake a highly nonlinear analysis, alternating between various aspects of the data as their relevance changes during the course of the analysis.

This linked-list structure partially reflects the way some data analysts think of the analysis process itself.

A somewhat simpler special case is that of a binary tree—an analysis in which every node represents a decision point between two possible continuations. Traversing a binary tree can be easily done using a *stack*, or a last-in, first-out queue. With such a structure each node has exactly one predecessor (except for the "root" node) and at most one successor. The only node without a successor is the last node created, which means it is at the "top" of the stack. One "leaves" a node in one of only two ways: by removing the node from the stack entirely and returning to its predecessor (thereby making the predecessor the top of the stack), or by creating a successor node. Simply having a workspace which allows automatic creation of a stack of worksheets may be sufficient for many purposes.

Whether the computing environment allows a full hierarchical tree structure or merely the ability to switch between named worksheets in an unstructured way, it is important to have the ability to copy subsets of the data from one worksheet to another. Note that this copying must have a "one-way" character in the case of the tree-structured workspace, since only nodes which have no children can be allowed to be modified.

5.6 Active Links

The previous section introduced the notion of linked worksheets or data sets, where the links represent the logical structure of the data analysis. There is another kind of linking, however, which links columns of a worksheet to a computation. These links are called *active links*, because changes in the worksheet cause immediate changes to be made either in the contents of other columns in the worksheet, or in the contents of an associated window, or both. The passive links of the workspace represent the relationship between two versions of the data set; an active link specifies a computational relationship among columns in the data set.

The idea of the active link is closely related to the notion of a "cell" in the many spreadsheet programs available for microcomputers. Each cell of the spreadsheet can contain either numerical data, or the description of a computation to be performed on other cells in the spreadsheet. In the latter case, the contents of the cell is automatically recomputed whenever the contents of the cells to which it refers is modified. The cells which appear in one of these computational formulas may themselves to be the results of other formulas. What the user sees is not the algebraic relationships among the cells, but rather the result of all of the algebraic computations. Changing any of the input variables causes automatic recomputation of the entire spreadsheet.

The idea is too useful not to be adapted to data analysis, particularly as it provides a useful alternative to macros or to replaying journal segments in repetitive, but local, computations. For example, consider the problem of performing a multiple regression fit, $Y = X\beta$, using the same Y and X variables, but omitting one or several points in succession. The point is to assess the effect of particular points on the analysis. This is cumbersome enough to do once in such systems as Minitab or GLIM, due in part to the inadequacies of the data structures built in to those systems. In Minitab, for instance, one can either PICK and JOIN rows to be retained into a new set of columns corresponding to the original columns, or (for single-point deletions) one can generate a column of row identification numbers and then OMIT a row with the selected identification number, again placing the result in a new set of columns, or one can generate a column of ones and zeros, zero representing points to be omitted, and then running a weighted regression with the dummy column as weights. After selecting the rows to be included (by whatever method), the appropriate regression command must be given, followed by the commands to construct and to display the residual plots.

The solution using active links is to associate a particular column with the dependent variable Y, particular columns with the regressors X, and another column with, say, the column of weights. These columns are then actively linked to two new columns, the contents of which are defined to be the residuals from the weighted regression of Y on X, and the fitted values from the same model. In addition, two windows are created the contents of which contain the regression statistics and the residual plot, respectively. Whenever the column of regression weights is modified (hence changing the subset of cases included in the regression), both the regression statistics and the residual plot are automatically recalculated and displayed. This makes it very easy for the analyst to perform one logical operation (change the subset of included cases) and then immediately see the effect on the matters of interest.

Another example, somewhat simpler, is that of selecting a transformation for the dependent variable Y in the regression above. The dependent-variable column can itself be replaced by an active link which raises Y to a power, where the power is a constant in the Minitab worksheet. Changing the value of this constant causes the dependent-variable column to be recomputed, which causes the regression and residual plots to be recomputed. By replacing the dependent-variable column by various transformed versions of Y in this way, the effect of a transformation on curvature in the residual plot can be instantaneously determined. Again, the data analyst can focus on one thing—twiddling the parameter of interest—while directly observing its effects. This facilitates the

comparisons necessary to choose between, say various candidate transformations, or various alternative subsets of the cases.

5.7 Side Computations, Filters, Tasks, and Pipes

In the middle of an analysis we often find that we would like to compute a function on a grid of points, where either the function, or the grid, or both depend on the current contents of the worksheet. More generally, we want to perform a *side computation* based on the current data, the results of which we would like to paste into the current worksheet for further display or analysis. Typically, these computations are the sort of thing we would like to perform in a programming language such as FORTRAN, not GLIM. Indeed, sometimes it is impossible to do the side computation within the statistical package itself.

For example suppose that, for each row of an $n \times p$ matrix the statistician wants to construct a *column* of length m. This task is exceedingly difficult, if not impossible, to accomplish using most packages. Yet this is precisely what must be done to construct the coordinates for Andrews (1972) plots, for example. (This example was brought to my attention by David Wallace.)

As another example, after fitting a nonlinear function of X to a dependent variable Y, it may be desirable to plot the original data with the estimated mean function overlayed. Since the grid of points on which the values of X fall in the data may be quite unevenly spaced, it is desirable to plot $\hat{Y}(x)$ on a narrow mesh, equally spaced grid of points the range of which is determined by that of the data.

Some ideas from the UNIX environment are useful in dealing with this problem. It is relatively easy in UNIX both to create a new *process* and to make the output from one process be the input to another using *pipes*. A process is a job or a task which consists of a running computer program. UNIX is an operating system designed to allow many different tasks, or processes, to run simultaneously; indeed, UNIX allows each user to create as many processes as he or she likes. More accurately, it allows for many different tasks to share the same resources of a computing system by scheduling resources and their use by tasks. Such computing systems are said to provide for *multitasking*. (Etymological note. While the terms "process" and "task" are synonymous, the terms "multiprocessing" and "multitasking" are not (although they are often confused). The former applies to a computing system which has several central processors which can be shared by tasks; the latter means that the system allows several tasks to contend for resources.) A program which accepts input from a standard device and which writes output to a standard device is said to be operating as a *filter*. In our case, we can think of the data worksheet as the "device" both from which input is read and into which output is to be written.

The data analysis environment should make it easy to leave the analysis program temporarily in order to write or to invoke a program to do the side computation. It should then be possible to return to the Minitab process and to invoke a Minitab command which does the following. It creates a new process which runs the filter program, and then it pipes the contents of the selected input portions of the worksheet to the filter, and pipes the filter's output into specified columns of the worksheet. When the filter process is complete, the process is killed.

5.8 Graphs

Much has been written recently concerning statistical graphics executed using computer graphics hardware and software, and new methods for statistical analysis based on the capabilities of computer graphics systems have been proposed. New graphical methods can incorporate animation and simulated motion, as well as dynamically adjustable color, as integral features. Most of the emphasis has been on designing useful new methods and on implementing them effectively on computing systems. (See for instance, Friedman and Stuetzle (1983) and Thisted (1984).) What has been little discussed are ways of integrating graphical methods of all kinds—both new and old—into effective strategies for data analysis using computer assistance. In short, the nature of computing environments which make it easier to create, retrieve, and compare graphical displays has not been systematically studied. Many of the principles of software design illustrated above apply equally well to graphics, and some of the specific suggestions for data analysis environments can be carried over directly to the graphical aspects of data analysis as well. Rather than discuss specific methods such as those involving kinematic displays, the remainder of this section will be devoted to a few comments suggesting how graphics can be integrated into the overall data analysis environment.

Graphic displays are essential elements of effective data analysis. A good computing environment will make it easier to work with graphic displays during the course of data analysis. Since much of the analysis process is based on comparing the consequences of alternative models, the environment should make it easy to compare similar graphic displays from competing models. For instance, we have already discussed a way to use active links in selecting an appropriate transformation of a response variable Y by continously varying a transformation parameter while monitoring the effect on a plot of Y against a suitable predictor X.

Using multiple windows, we can display two or more graphs side by side. To do this, the computing environment should make it easy to save and to retrieve graphs that are created in the course of an analysis. If the metaphor for the electronic notebook is the three-ring binder, an appropriate metaphor for saving and retreiving graphs is the slide carousel. Each graph can be thought of as a 35-mm slide, which can be inserted into, or retrieved from, a "carousel." One can review and compare graphs by cycling through the carousel, "projecting" the graph in a window devoted to that purpose.

An interesting and instructive description of an attempt to integrate graphics and graphical methods into a computing environment, in this case, an environment for economists and econometricians, is described in Williams (1984).

6. COMPUTING ENVIRONMENTS AND EXPERT SYSTEMS

A good computing environment for data analysis is easy and natural to use, and makes it easier for novices to take advantage of techniques and strategies discovered and refined by more experienced statisticians. It is a small conceptual step from the idea of a useful environment to that of a *helpful* one. An expert system for data analysis would consist of software which could offer consultation of various sorts during the course of an analysis session. Ideally, it would be fully integrated into the computing environment.

We have discussed ways in which a computing environment could be designed, and we have emphasized that such a computing system should be nearly transparent to the user, in the sense that the things that are easy to do within the system are precisely the things that the user would actually like to do to get the job done. To achieve this goal, it is necessary to understand what it is that data analysts—especially the best ones—actually do. Indeed, the procedures that statisticians employ and the strategies they adopt reflect experience and understanding of data sets, of statistical methods, of mathematical structures, and of the nature of experimentation and data collection (Thisted, 1985a). This base of knowledge is the starting point for constructing an expert system to provide assistance (or even just a second opinion) to users of a statistical system.

In order to develop a base of statistical knowledge that could be incorporated into an expert system, a logical starting point is to collect transcripts of data analysis sessions, to observe precisely what it is that analysts do. These transcripts, automatically collected by the computing environment described here, could then be examined and analyzed as records of the data analysis process. The authors of these transcripts could be interviewed about their reasons for doing particular things, or doing them in a particular order. Through these devices the heuristics that are used—whether conscious or not—and their effectiveness can be assessed. Thus, an appropriately designed computing environment can serve as a useful tool for constructing the knowledge base of an expert statistical consultant. The dual role of computing environments in representing and discovering knowledge of statistical analysis strategies is discussed in Thisted (1985b).

7. OTHER COMPUTING ENVIRONMENTS FOR STATISTICIANS

Statisticians do far more than data analysis, and the role of the computer in statistics is far larger than simply that of performing the computations involved in analyzing data. These other tasks that involve computing could themselves profit from suitably designed computing environments. Of course these environments would differ greatly from the data analysis environment considered here, simply because the tasks which should be easy and natural are so different in the different settings. We outline some areas of statistical endeavor in which designing a computing environment could well produce great benefit.

The "Monte Carlo Workbench" would consist of a large collection of tools and building blocks necessary to conduct a simulation study. The tools would include a set of random number generators, looping constructs, automatic performance monitoring modules, and interfaces which produce output suitable for analysis using a data analysis package. This environment would make it easy to construct pilot studies, perhaps even full simulation studies, using standard experimental designs and producing easily examined and analyzed results.

The "Theoretical Statistician's Environment" would include such things as numerical and symbolic integrators and differentiators, and plotting routines. Such an environment would make it easy to explore such diverse things as asymptotic expansions, operating characteristics, and likelihood surfaces.

Just as the statistical methodologist must often employ simulations to explore numerical statistical methods, so must the investigator of graphical methods also explore the range of situations over which particular candidate methods perform well. An environment for constructing and testing new graphical methods would play much the same role for such an investigator as the Monte Carlo Workbench might play for those studying more traditional methods.

ACKNOWLEDGMENTS

This paper is based upon work supported in part by National Science Foundation Grant DMS-8412233 to

the University of Chicago. I am grateful to David Andrews, Michael Meyer, and David Draper for stimulating discussions which influenced this paper, and to the Executive Editor and referee for helpful comments.

REFERENCES

ANDREWS, D. F. (1972). Plots of high-dimensional data. *Biometrics* **28** 125–136.

BECKER, R. A. and CHAMBERS, J. M. (1984a). *S: An Interactive Environment for Data Analysis and Graphics.* Wadsworth, Belmont, Calif.

BECKER, R. A. and CHAMBERS, J. M. (1984b). Design of the *S* system for data analysis. *Comm. ACM* **27** 486–495.

BECKER, R. A. and CHAMBERS, J. M. (1985). *Extending the S System.* Wadsworth, Belmont, Calif.

COWLEY, P. J. and WHITING, M. A. (1985). Managing data analysis through save states. In *Computer Science and Statistics: Seventeenth Symposium on the Interface* (D. M. Allen, ed.). North-Holland, Amsterdam, in press.

DONOHO, D. L. (1983). *DART: A Tool for Research in Data Analysis.* Ph.D. thesis, Department of Statistics, Harvard Univ.

FRIEDMAN, J. H. and STUETZLE, W. (1983). Hardware for kinematic statistical graphics. In *Computer Science and Statistics: Proceedings of the Fifteenth Symposium on the Interface.* (J. E. Gentle, ed.) 163–169. North-Holland, Amsterdam.

GALE, W. A. (1985). Knowledge representation in statistics. In *Computer Science and Statistics: Seventeenth Symposium on the Interface* (D. M. Allen, ed.). North-Holland, Amsterdam, in press.

JOY, W. and GAGE, J. (1985). Workstations in science. *Science* **228** 467–470.

MCCULLAGH, P. and NELDER, J. A. (1983). *Generalized Linear Models.* Chapman and Hall, London.

MCDONALD, J. A. and PEDERSON, J. (1985a). Computing environments for data analysis. I. Introduction. *SIAM J. Sci. Statist. Comput.* **6** 1004–1012.

MCDONALD, J. A. and PEDERSON, J. (1985b). Computing environments for data analysis. II. Hardware. *SIAM J. Sci. Statist. Comput.* **6** 1013–1021.

NELDER, J. A. and WEDDERBURN, R. W. M. (1972). Generalized linear models. *J. Roy. Statist. Soc. Ser. A* **135** 370–384.

RYAN, T. A., JR., JOINER, B. L. AND RYAN, B. F. (1976). *MINITAB Student Handbook.* Duxbury, North Scituate, Mass.

THISTED, R. A. (1984). An appraisal of statistical graphics. In *Statistics: An Appraisal, Proceedings of the Fiftieth Anniversary Conference, Iowa State Statistical Laboratory* (H. A. David and H. T. David, eds.) 605–624. Iowa State University Press, Ames.

THISTED, R. A. (1985a). Knowledge representation for expert data analysis systems. In *Computer Science and Statistics: Seventeenth Symposium on the Interface* (D. M. Allen, ed.). North-Holland, Amsterdam, in press.

THISTED, R. A. (1985b). Representing statistical knowledge and search strategies for expert data analysis systems. In *Artificial Intelligence and Statistics* (W. A. Gale, ed.). Addison-Wesley, Reading, Mass., in press.

WILLIAMS, T. L. (1984). A graphical interface to an economist's workstation. *IEEE Comp. Graphics Applications* **4** 42–47.

Comment

John M. Chambers

Statisticians have traditionally shown a wide range of attitudes to computing, from being deeply involved and enthusiastic to regarding the subject with some distaste, or at least suspicion. In the early days negative attitudes were often strong. As an enthusiastic undergraduate in the early 1960s, I was cautioned by a faculty advisor that, while computers certainly had their uses, actually programming them was not really compatible with a research career.

Current attitudes usually exhibit less outright opposition than feelings of confusion over hardware and software choices, combined with some resentment at the learning and relearning that each new development seems to require. Professional statisticians reasonably want to know how they can benefit from computers in their work. If an overview such as Professor Thisted's paper could clarify this issue, it would do its readers a service. In the present case, I worry that several helpful insights may have been somewhat buried by irrelevant details, shifts in viewpoint and unnecessarily old-fashioned examples. The reader's overall level of confusion may rise rather than fall. Rather than quibbling with the paper, however, it will be more helpful for me to present, briefly, my own view of the topic.

What *are* the important points about computing environments for data analysis? Here are two, from which most of the relevant conclusions follows:

(1) Computing environments should be judged by their complete, present and future, contribution to their user's effectiveness.

(2) Most of the important improvements in statistical computing environments have come through advances in general computing, not from anything statisticians have done. This will continue to be true for the immediate future.

Point (1) implies that it is not sufficient to ask how easily the user can carry out a specific current data analysis, important as that question may be. Two other questions must also be weighed. How well does the environment carry out the nonstatistical tasks

John M. Chambers is Head of the Statistics and Data Analysis Research Department, AT&T Bell Laboratories, Murray Hill, New Jersey 07974-2070.