SLAC–PUB–3577
STAN–LCS–09
February 1985
(M)

COMPUTING ENVIRONMENTS FOR DATA ANALYSIS*

Part 1: Introduction

John Alan McDonald

and

Jan Pedersen

Department of Statistics

and

Stanford Linear Accelerator Center

Stanford University, Stanford, California 94305

ABSTRACT

This is the first in a series of papers on aspects of modern computing environments that are relevant to statistical data analysis. We argue that a network of graphics workstations is far superior to conventional batch or time-sharing computers as an environment for interactive data analysis. The first paper in the series provides a general introduction and motivation for more detailed considerations of hardware and software in subsequent parts.

Portions of this paper presented at the 17th Symposium on the Interface of Computer Science and Statistics, Lexington, Kentucky, March 17-19, 1985.

---

* Work supported in part by the Department of Energy, contract DE-AC03-76SF00515, by the NSF Mathematical Sciences Postdoctoral Research Fellowship, by the Office of Naval Research contract N00014-83-K-0472 and grant N00014-83-G-0121, and by the U.S. Army Research Office contract DAAG29-82-K-0056.

Chapter 1

Introduction

Statistics has long been thought of as *applied mathematics*. Certain parts of it, especially data analysis, could easily be viewed as *applied computation*. In this context, different research issues assume importance, in particular, the design and implementation of *computing environments for data analysis*.

Analyzing data requires computing, whether with paper-and-pencil or with a Cray super computer. The *computing environment* determines what sorts of statistical methods are practical. More importantly, the statistician's unconscious assumptions, or *mental model* of the computing environment determines the kinds of new statistical methods that are likely to be invented.

Most current research in statistical computing is based on a *batch processing* model of computing environments that was appropriate twenty years ago. With a few exceptions, statisticians have not addressed the implications current and future developments in scientific computing environments.

Although no single number can be a complete summary, it is probably fair to say—for purposes of discussion—that the performance per dollar of computing equipment has increased by a factor of a thousand over the past twenty years. The large quantitative change implies a qualitative change in how computers can be used. For comparison, the magnitude of the difference is larger than that between walking and flying—if one could fly for the same price as one could walk.

This is the first part of a series of papers that discuss *local networks* of *graphics workstations* as environments for statistical computing. In this, the first part, we provide general background and motivation. Future parts will describe relevant aspects of computing hardware and software, in particular present and future scientific programming environments.

Chapter 2

Trends in Scientific Computing Environments

2.1 Pencil and Paper

Most statistical methods originated in the pencil-and-paper computing environment of 50 years ago (perhaps we should say the pencil-paper-statistical-table-hand-calculator environment) and were designed with its particular constraints and limitations in mind. Although some statisticians were prodigiously good at arithmetic, there was an obvious practical limit on the volume of computation feasible for a given analysis. This implied statisticians were restricted to fairly small data sets and simple methods requiring relatively little computing, by today's standards. The result was an emphasis on idealized models for data whose primary motivation was the availability of closed form solutions or convenient asymptotic approximations.

Balanced against these shortcomings was the flexibility of the pencil-and-paper environment. Since the statistician was immediately involved in every stage, the analysis could be easily adapted to take advantage of contextual information and common sense. In addition, the time between the conception of a qualitatively new method and its implementation was essentially zero.

2.2 Batch Processing

Twenty years ago, when computing machinery was a scarce resorce, great effort went into making efficient use of machine time at the expense of the time of those who used it. A university computer center would typically have a single *mainframe* computer (e.g. an IBM 360), which could easily be in the million dollar range. To use the computer, users submitted programs, usually in the form of decks of punched cards, to the computer's job queue. A batch operating system controlled the execution

of jobs, typically processing one job at a time, from start to finish. The results appeared, hours or days later, on reams of awkwardly sized line-printer paper. We will refer to this type of computing environment as *batch processing*.

Although batch computing environments are, fortunately, nearly extinct (at least for scientific computing), it is important to keep the batch processing model (punch card to batch processor to line printer) in mind. Not only are the standard statistical packages cast in this form, for example SAS, BMDP , and SPSS; much of state-of-the-art statistical computing has this model as an implicit underlying assumption.

The batch processing model limits the sort of software that can be developed. The two most confining aspects of batch processing are the long turn-around time for the simplest computations and the even longer time to make minor changes to a program. The result is a style of data analysis based on the concept of a *statistical package*, a program, or collection of programs, implementing a small set (10 to 50) of fixed operations that can be applied to sets of data.

Because the time to modify a program is so long, each operation is essentially fixed, with perhaps a few options that can be set before execution. The set of provided operations in a package is usually small and new operations are difficult or impossible to add. There is, therefore, a tendency to force data sets of great individuality into models that are not really appropriate, just because they happen to be in the available packages. Prior knowledge, both about complex internal structure and about the external context of the data, must often be ignored. The lack of flexibility is pervasive and imposes severe limits on the creativity in solving the new problems posed by each new data set.

Because the execution turn-around time is so long, each operation is designed as a more-or-less complete analysis. Intermediate choices in an analysis must be automated; they cannot be based on interpretations of intermediate results. These automated decisions are often a poor substitute for staightforward interactive methods. As a result, it is not uncommon for packaged routines to attempt to anticipate every possible contingency, generating volumes of output of which only some small part is actually relevant.

An argument can be made that data analysis has regressed by entering the computer age. Although the computing constraint has been largely eliminated, statisticians have lost the flexibility inherent in the pencil-and-paper mode of analysis. John Tukey drives this point home in the postscript to EDA, a book written almost entirely from the pencil-and-paper point of view.

> This book focusses on paper and pen(cil) — graph paper and tracing paper when you can get it, backs of envelopes if necessary — multicolor pen if you can get it, routine ballpoint or pencil if you cannot. Some would say that this is a step backward, but there is a simple reason why they cannot be right: much of what we have learned to do to data can be

> done by hand — with or without the aid of a hand-held calculator– *LONG BEFORE* one can find a computing system — to say nothing of getting the data entered. Even when every household has access to a computing system, it is unlikely that "just what we would like to work with" will be easily enough available. Now — and in the directly forseeable future — there will be a place for hand calculation. [Tukey, 1977, p. 663]

The challenge before us is to recover the flexibility that has been lost, but also to augment it with the considerable computing power that should be available at our finger tips.

2.3 Time-sharing

As computer prices dropped there was less concern for efficient use of machine time and more concern for the efficient use of people time. This led to the development of *time-sharing operating systems* and a trend towards smaller computers (which were often only smaller in the sense that they were owned and used by a smaller number of people).

A time-shared mainframe communicates with its users through many remote terminals and provides a few centralized peripheral devices (e.g. a line printer, disk drive, tape drive, etc.). The mainframe runs a multi-user operating system and users rent time, sharing the central processor.

Some statistical packages, e.g. Minitab, take advantage of time shared computing to provide some degree of interaction, but most statistical computing has changed only to the extent that it is easier to submit batch jobs.

Statistical programming languages, like S [Becker and Chambers, 1984a, 1984b] and ISP [Donoho, Huber, Ramos, and Thoma, 1982], make better use of the possibilities of interactive computing. A *statistical language* provides tools for combining primitive functions and data structures to create new, higher-level functions and data structures. A statistical language is therefore extensible, in a way that a statistical package is not. That is, new operations can be defined and easily integrated into the language. In a package the procedures stand alone and communication between procedures is awkward or impossible, while in a language the operations are designed to work together.

2.4 Networks of Graphics Workstations

Networks of graphics workstations are the logical next step in providing a friendly and more powerful computer environment.

We begin with a few definitions:

A *workstation* is a complete computer that is used by a single person. The workstations discussed in this paper may be thought of as roughly equal—in speed of computation, amount of memory, and so forth—to a VAX (somewhere in the range covered by the 11/730 to 11/790). In other words, they do not have the limitations that one might associate with present day personal or home computers.

A *graphics workstation* includes, in addition, a high resolution *bitmap* display and *graphical input devices* (e.g. mouse, trackerball, joystick) as integral parts [Beatty, 1983; Foley and VanDam, 1982; Newman and Sproull, 1979]. This combination permits a natural, graphical language for communication between user and computer, which, in turn, leads to the design of computing environments that are more powerful, more sophisticated, and easier to use.

Examples are workstations built by Apollo, Sun, Chromatics, Symbolics, Ridge, and Masscomp, which are discussed in more detail in part 2 of this paper. Briefly, some of the properties of a graphics workstation appropriate for use in statistics are:

- Hardware
  - A central processor equivalent to a Vax (500,000 to several million instructions per second).
  - Fast floating point equivalent to a Vax (500,000 to several million floating point operations per second).
  - Several megabytes of physical memory (RAM).
  - virtual memory (a large address space).
  - Tens to hundreds of megabytes of disk storage.
  - Bitmap graphics display(s), at a resolution of approximately 1000x1000, in black-and-white and/or color
  - Auxiliary processors, including array processors for rapid numerical computation and graphics processors for fast picture drawing.
  - Graphical input devices such as a mouse.
  - Both hardware and software to support communication over a high speed (several megabits per second) local network (usually some flavor of Ethernet).
- Programming Environment

  A user controls the action of a computer through a set of (software) tools—*the programming environment*. We are intentionally blurring the distinction between direct, interactive (interpreted) commands and the more complex deferred instructions produced by writing, compiling, linking, and loading a program. The basic alternatives in graphics workstations are a conventional operating system,

such as Unix [Deitel, 1983; Kernighan and Mashey, 1981; UNIX, 1978], or an integrated programming environment, such as Interlisp-D [Sheil, 1983; Teitleman and Masinter, 1981] or Smalltalk [Goldberg, 1983, 1984; Goldberg and Robson, 1984; Krasner, 1983;]. Examples of facilities provided by a programming environment are [Deutsch and Taft, 1980]:

- Programming languages.
- A file system.
- An (intelligent) display editor.
- Interactive debugging tools.
- A windowing system for effective use of the bitmap display.
- Support for graphical interaction.

A *network* is a combination of hardware and software that permits independent workstations and their users to communicate—in other words, transfer data—rapidly and share resources (such as tape drives, printers, etc.) [Green, 1982; Tannebaum, 1981a, 1981b]. We are most concerned here with *local networks* (connecting computers within a building or a campus), in contrast to *wide area networks* (connecting machines accross the country or around the world).

In addition to graphics workstations, the network might include some of the following:

- Small personal computers, such as the Apple Macintosh.
- Several-user midi-computers, such as a VAX 11/750.
- Many-user mainframes, such as DEC 20.
- Special purpose number crunching engines, such as a Floating Point Systems array processor.
- Super computers, such as a Cray.
- Printers
- Fileservers—computers with large amounts (at least several gigabytes) of disk storage used to store files and databases shared by some or all the machines in the network.
- Tape drives
- Gateways to other local networks.

- Gateways to wide-area networks, such as Arpanet.

The distinction between personal computers, graphics workstations, and midi-computers may disappear as graphics workstations become more powerful than midi-computers and no more expensive than personal computers.

It is important to emphasize that this series of papers describes computing environments that will not be fully realized for five to ten years. Although graphics workstations are commercially available, they range in price from $15,000 to $150,000, which is too expensive for there to be one on every desk. Also, even if the hardware is available, software for statistical applications has yet to be written.

Another serious problem is the lack of standardization of network protocols and internal software that is necessary to allow communication and portability of software between machines of different manufacturers.

There are good reasons to expect the situation to change rapidly. For example, the recently announced Apple Macintosh has many of the desirable features of graphics workstations. However, it has a list price of only $2,500 (and is available in many universities for about $1,000).

Chapter 3

Why Workstations?

3.1 More for the money

If, over the past twenty years, the price of computer hardware has dropped by a factor of 1000, then we should be able to get 1000 times more computing for our dollar. There are two basic alternatives for taking advantage of the price decline: make the central mainframe 1000 times as big or make 1000 copies of the central mainframe. Each has advantages for different purposes.

For example, a single *super* computer can handle very large problems—in ways that 1000 smaller ones cannot.

On the other hand, the change in price is not as simple as a uniform 1000 fold drop in the cost of all aspects of computers. Non-linearities in price/performance give smaller computers great advantages for many uses.

The primary reason for the drop in the price of computer hardware has been the development of cheap large scale integrated circuits, which, in particular, has lead to cheaper processors and memory. A single-user workstation is less complex and can take greater advantage of VLSI technology than a multi-user machine, resulting in more instruction cycles per dollar. Furthermore, the drop in memory prices has made the combination of workstation and high quality, bitmap graphics displays affordable.

Single user machines also tend to have less software overhead; the effort normally committed to managing the complexity of a multi-user system can instead be invested in enriching the programming environment for the single user.

3.2 Graphical interaction

A basic premise of research in computing environments is that using a computer should be a natural activity, in the same sense that driving a car is natural.

An important advantage of workstations is the possibility of a natural, graphical

language for control of the computer. Bitmap displays and graphical input devices provide a "high bandwidth channel" (rich and fast information transmission both ways) for communication between person and machine.

Graphical interaction permits the design of sophisticated programming environments that are both easier for the novice and more powerful for the expert. We will discuss programming environments in more detail in a future part of the paper.

Acceptable graphical interaction is difficult to achieve on anything but a single user graphics workstation because it requires

- computing power on demand
- high speed data transfer between cpu and display

to have guaranteed fast response. On a multi-user machine, the response time will vary as the operating system varys the size of the time-slice given to each user. Multi-user machines typically communicate with graphics terminals at rates of about 10,000 bits per second; required communication speeds may be one or two orders of magnitude higher.

3.3 Local Control

We expect to see a graphics workstation on the desk of every statistician who wants one—in five to ten years. At present, the type of workstation we are considering is too expensive ($15,000 to $150,000) for that, but it is reasonable to expect a Statistics department to be able to buy one or more. There are a number of advantages that come from having a computer that belongs to a small group of people with similar interests and serves only one person at a time.

A single user computer has its total computing power available on demand. Among other benefits, this permits guaranteed response time for interaction. On a large mainframe the central processor's attention is divided among several users in a round-robin fashion so that in any given second no one user receives more than a fraction of a second of processing time, and that fraction depends on the numbers of other users currently active on the system.

The owner(s) of a workstation decides how to use its resources, not a large bureaucracy. This makes it possible to provide essentially free computer time for computer intensive methods; once the machine is paid for the owner(s) need not worry about incremental charges for cpu time.

Since a workstation is dedicated to a single user, it can be taylored to a specific application, e.g. data analysis, rather than maintaining compromises unavoidable in a general purpose computing environment.

3.4 New Statistical Methods

We have mentioned above two measures of performance of computing environments: execution turn-around time and program modification time. The quantitative change in program modification time (from hours or days in batch environments to seconds in sophisticated, integrated programming environments) has great implications which we will explore more fully in future parts of this paper.

In this section, we give some examples of how quantitative change in execution turn-around time leads to qualitatively new statistical methods.

Consider the length of time it takes to draw a scatterplot, with a line fit to the data by least squares. With pencil-and-paper, it might take days for moderate sized (500 observations) data sets. In a batch processing environment, it will take hours. In a timesharing environment, it will take minutes. On a graphics workstation, it need only take 1/10 of a second.

The fact that a new plot can be drawn in fractions of a second makes possible two new methods for data analysis that have been explored with prototype data analysis systems:

- Prim-9 [Fisherkeller, Friedman, and Tukey, 1974].
- Prim-H [Donoho, Huber, and Thoma, 1981; Donoho, Huber, Ramos, and Thoma, 1982].
- Orion I [Friedman and Stuetzle, 1982c; Friedman, McDonald, and Stuetzle, 1982; McDonald, 1982a, 1982b, 1982c].

3.4.1 Three-dimensional Scatterplots

There are a variety of techniques for drawing three-dimensional scatterplots [Nicholson and Littlefield, 1982; Littlefield and Nicholson, 1982]. The **Prim** and **Orion** systems use the human ability to percieve shape from motion. Essentially, a two-dimensional orthogonal projection of the three- dimensional point cloud is displayed on the graphics screen. Rotating the point cloud in real-time allows the user to view from any direction and gives a convincing and accurate perception of the shape of a three-dimensional scatterplot from the apparent parallax in the motion of the points. This is demonstrated in the **Prim-9** and **Orion** films [Prim-9, 1974; McDonald, 1982a, 1982b, 1982c].

Real-time motion is a demanding task. As is in the cinema, the apparently moving picture is actually composed of a sequence of static frames. Conventional movies are shown at 24 frames per second; television at 30 frames per second. Three-dimensional scatterplots can be successfully displayed somewhat more slowly—down to perhaps 3

frames per second. However, shape is perceived with more comfort at rates closer to 24 or 30 frames per second.

The rate at which motion graphics can be displayed is determined by:

- the time it takes to compute a new frame.
- the time it take to erase the old frame and draw the new one.

To display three-dimensional scatterplots—or other type of motion graphics–at 30 frames per second, all the computations associated with each frame must be completed in roughly 1/60 of a second, to allow time for drawing and erasing the frames. Rotation about an arbitrary axis with perspective [Foley and VanDam, 1982; Newman and Sproull, 1979] requires at least nine multiplications, nine additions, and two divisions—20 arithmetic operations per point. Therefore, for smooth rotation of a scatterplot with 1000 points, the computer may need to do 1,200,000 arithmetic operations per second. Orthogonal rotation about either the vertical or horizontal axis takes 3 multiplies and 2 adds per point; at a minimal rate of 5 frames per second, 1000 points demands about 50,000 arithmetic operations per second.

Moving scatterplots are usualy displayed by drawing and erasing a small symbol for each point. Such symbols may consist of from 1 to 100 pixels (picture elements). For five pixel symbols, erasing and re-drawing a frame with 1000 points in 1/60 second requires a graphics device that can do 600,000 pixel writes per second. Changing a pixel typically means sending 32 bits of address and 8 bits of data—a total of 5 bytes—from the computer to the graphics device. Therefore smooth rotation may require the computer to be able to communicate with the graphics device at 3 megabytes per second.

This type of real-time motion is impractical on a time-shared computer, since one cannot rely on a sufficiently large timeslice to do the necessary computing. In contrast, a workstation is dedicated to a single user, so the timing constraints are more easily satisfied. Also, real-time motion is inhibited in time-shared systems by slow data transfer between a central processor and a remote graphics terminal.

3.4.2 Interactive Model Fitting

Suppose we want to fit a model with several parameters to a set of data. Also suppose we can draw a picture that shows how well the model fits for any given values of the parameters. Then it may be useful to be able to vary some or all of the parameters interactively, with a graphical input device (such as mouse, touchpad, trackerball, joystick).

A simple example is linear regression with power transformations. That is, we approximate observed data, $\{x_i, y_i\}$, with the model:

$$y = a + b \cdot x^{\theta}.$$

For fixed values of θ, $\hat{a}(\theta)$ and $\hat{b}(\theta)$ are fit by least squares. On a graphics workstation, we can use a dial, or some equivalent *graphical input device* to control the value of θ. We would display a scatterplot of y_i versus $z_i = x_i^\theta$ with the least squares line: $y = \hat{a}(\theta) + \hat{b}(\theta) \cdot z$. The picture must change smoothly as we move the dial that controls θ. To do this the graphics system must be able to compute the transformation, $z_i = x_i^\theta$, the least squares fit, and erase and redraw the picture about 30 times a second.

A still more demanding example is Interactive Projection Pursuit Regression, which is described by Friedman and Stuetzle [1981b, 1982a, 1982b] and McDonald [1982a] and demonstrated in the film by McDonald [1982c].

Chapter 4

Stay tuned til next week

This has been an introduction to a series of papers on research in computing environments for data analysis. It is intended to provide an overview and motivation for what follows.

There are two important aspects to computing environments. The more fundamental and limiting of the two, *Hardware,* will be reviewed in detail in part 2. The other aspect, *Programming Environments*, is perhaps of more immediate interest as an area of research for statisticians. It will be the basis for parts 3 through 5 of this paper:

- Issues in the Design of Programming Environments for Data Analysis
- Conventional Programming Environments: Unix.
- Integrated Programming Environments: Lisp and Flavors

Chapter 5

References

Barstow, D.R., Shrobe, H.E., and **Sandewall, E.,** eds. (1984)
Interactive Programming Environments
New York, McGraw Hill, 1984
Beatty, J.C. (1983)
Raster Graphics and Color,
The American Statistician, Vol. 37, Num. 1, Feb. 1983, pp 60-75
Becker, R.A. and **Chambers, J.M.** (1984a)
Design of the S System for Data Analysis
Communications of the ACM, Vol. 27, Num. 5, May 1984, pp 486-495
Becker, R.A. and **Chambers, J.M.** (1984b)
S: An Interactive Environment for Data Analysis and Graphics
Belmont, CA., Wadsworth, 1984
Deitel, H.M. (1983)
An Introduction to Operating Systems.
Reading, Mass., Addison Wesley, 1983
Deutsch, P. and **Taft, E.,** eds. (1980)
Requirements for an Experimental Programming Environment,
Xerox PARC Report CSL-80-10, Palo Alto, CA.
Donoho, D., Huber, P.J., and **Thoma, H.** (1981)
The Use of Kinematic Displays to Represent High Dimensional Data,
Computer Science and Statistics: Proc. of the 13th Symposium on the Interface.
Edited by Eddy, W.F., New York, Springer-Verlag, 1981
Donoho, D., Huber, P.J., Ramos, E. and **Thoma, H.** (1982)
Kinematic Display of Multivariate Data,
Proc. of the Third Annual Conference and Exposition of the National Computer Graphics Association, Inc., Volume I.
Fisherkeller, M.A., Friedman, J.H., and **Tukey, J.W.** (1974)
Prim-9, An Interactive Multidimensional Data Display and Analysis System, // Proc.

of Pacific 75, ACM Regional Conference.
Foley, J.D. and **Van Dam, A.** (1982)
Fundamentals of Interactive Computer Graphics.
Reading, Mass., Addison Wesley, 1982
Friedman, J.H., McDonald, J.A., and **Stuetzle, W.** (1982)
An Introduction to Real Time Graphical Techniques for Analyzing Multivariate Data
Proc. of the Third Annual Conference and Exposition of the National Computer Graphics Association, Inc., Volume I.
Friedman, J.H. and **Stuetzle, W.** (1981b)
Projection Pursuit Regression
JASA, vol. 76, pp 817-823
Friedman, J.H. and **Stuetzle, W.** (1982a)
Smoothing of Scatterplots
Dept. of Statistics Tech. Rept. Orion 3, Stanford University.
Friedman, J.H. and **Stuetzle, W.** (1982b)
Projection Pursuit Methods for Data Analysis
in *Modern Data Analysis*
Launer, R.L. and Siegel, A.F., eds. New York, Academic Press, 1982
Friedman, J.H. and **Stuetzle, W.** (1982c)
Hardware for Kinematic Statistical Graphics
Computer Science and Statistics: Proc. of the 15th Symposium on the Interface.
Goldberg, A. (1983)
The Influence of an Object-Oriented Language on the Programming Environment
reprinted in **Barstow, D.R., Shrobe, H.E.,** and **Sandewall, E.,** eds. (1984)
Interactive Programming Environments
Goldberg, A. (1984)
Smalltalk-80: The Interactive Programming Environment
Reading, Mass., Addison Wesley, 1984
Goldberg, A. and **Robson, D.** (1983)
Smalltalk-80, The Language and Its Implementation
Reading, Mass., Addison Wesley, 1983
Green, P.E., ed. (1982)
Computer Network Architectures and Protocols
New York, Plenum Press, 1982
Kernighan, B.W. and **Mashey, J.R.** (1981)
The UNIX Programming Environment
Computer Vol 14, Num 4, April 1981, pp. 25-34
Kernighan, B.W. and **Ritchie, D.M.** (1978)
The C Programming Language
Englewood Cliffs, N.J., Prentice Hall, 1978
Krasner, G., ed. (1983)

Smalltalk-80, Bits of History, Words of Advice
Reading, Mass., Addison Wesley, 1983
Littlefield, R.J. and **Nicholson, W.L.,** (1982)
Use of Color and Motion for the Display of Higher Dimensional Data
Proc. of the 1982 DOE Statistical Symposium.
McDonald, J.A. (1982a)
Interactive Graphics for Data Analysis
Ph.D. thesis, Dept. of Statistics, Stanford; available as *Dept. of Statistics Tech. Rept. Orion 11, Stanford University.*
McDonald, J.A. (1982b)
Exploring Data with the Orion I Workstation
a 25 minute, 16mm sound film, which demonstrates programs described in McDonald (1982a). It is available for loan from: Jerome H. Friedman, Computation Research Group, Bin # 88, SLAC, P.O. Box 4349, Stanford, California 94305
McDonald, J.A. (1982c)
Projection Pursuit Regression with the Orion I Workstation
a 20 minute, 16mm color sound film, which demonstrates programs described in McDonald (1982a). It is available for loan from: Jerome H. Friedman, Computation Research Group, Bin # 88, SLAC, P.O. Box 4349, Stanford, California 94305
Newman, W.M. and **Sproull, R.F.** (1979)
Principles of Interactive Computer Graphics, 2nd ed.
New York, McGraw Hill, 1979
Nicholson, W.L., and **Littlefield, R.J.** (1982)
The Use of Color and Motion to Display Higher Dimensional Data
Proc. of the Third Annual Conference and Exposition of the National Computer Graphics Association, Inc., Volume I.
Prim-9 (1974)
Prim-9, a 30 minute, 16mm sound film, It is available for loan from: Jerome H. Friedman, Computation Research Group, Bin # 88, SLAC, P.O. Box 4349, Stanford, California 94305
Sheil, B.A. (1983)
Power Tools for Programmers
reprinted in **Barstow, D.R., Shrobe, H.E.,** and **Sandewall, E.,** eds. (1984)
Interactive Programming Environments
Tannenbaum, A.S. (1981a)
Computer Networks
Englewood Cliffs, N.J., Prentice Hall, 1981
Tannenbaum, A.S. (1981b)
Network Protocols
ACM Computing Surveys, Vol. 13, Num. 4, Dec. 1981, pp 453-489
Teitleman W. and **Masinter, L.** (1981)

The Interlisp Programming Environment
reprinted in **Barstow, D.R., Shrobe, H.E.,** and **Sandewall, E.,** eds. (1984)
Interactive Programming Environments
Tukey, J.W. (1977)
Exploratory Data Analysis
Reading, Mass., Addison Wesley, 1977
UNIX (1978)
special issue on Unix, *The Bell System Technical Journal, Vol 57*