SLAC-253
STAN-ORION 011
UC-32
(M)

INTERACTIVE GRAPHICS FOR DATA ANALYSIS*

JOHN ALAN MCDONALD

Stanford Linear Accelerator Center
Stanford University
Stanford, California 94305

August 1982

Prepared for the Department of Energy

under contract number DE-AC03-76SF00515

* Ph.D. Dissertation.

ABSTRACT

This paper reports on work in a new branch of research in statistical methods: applications of interactive computer graphics. I describe several programs written for the Orion I workstation, an experimental computer graphics system built at the Stanford Linear Accelerator Center in 1980-81. These programs demonstrate new methods for data analysis made possible by advances in microprocessor and computer graphics technology. These methods make extensive use of color, real-time motion, and interaction to discover structure in many-dimensional data. Two approaches to graphical analysis of many-dimensional data are discussed in detail: interactive projection pursuit and simultaneous, multiple views.

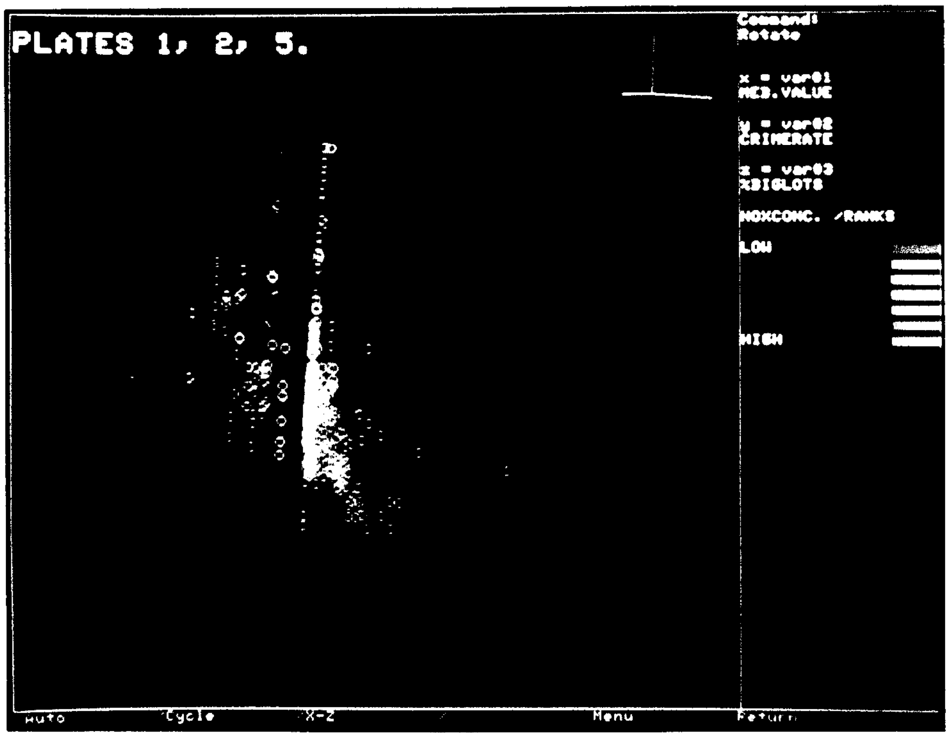
ACKNOWLEDGEMENTS

I would like to thank my advisors, Werner Stuetzle and Jerry Friedman, who designed Orion I, and who got me into this business. I thank Dave Parker for general wizardry. I thank the other members of the Computation Research Group at the Stanford Linear Accelerator Center who helped with many suggestions and in maintaining both hardware and software, including Dan Asimov, Andreas Buja, Mary Anne Fisherkeller, Dave Gustafson, Trevor Hastie, and John Steffani.

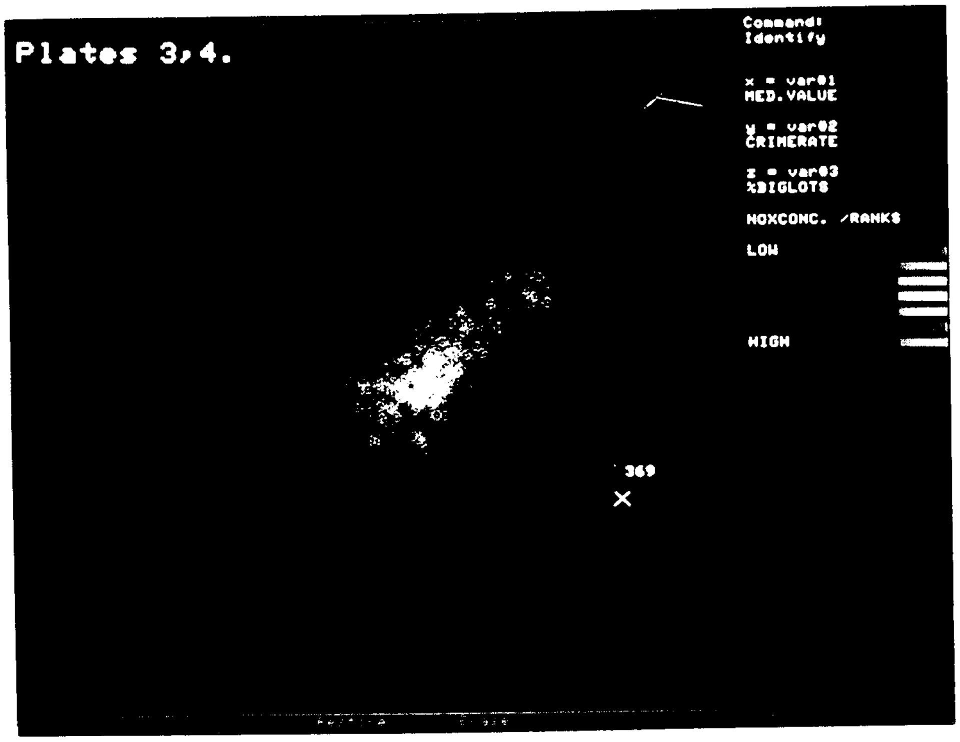
This research was supported in part by the Department of Energy under contracts DE-AC03-76SF00515 and DE-AT03-81-ER10843, the Office of Naval Research under contract ONR N00014-81-K-0340, and the U.S. Army Research Office under contract DAAG29-82-K-0056.

CONTENTS

LIST OF FIGURES


PLATES 1, 2, 5.
Command:
Rotate
x = var01
MED.VALUE
y = var02
CRIMERATE
z = var03
%BIGLOTS
NOXCONC. /RANKS
LOW
HIGH
Auto
Cycle
X-Z
Menu
Return

A


Plates 3,4.
Command:
Identify
x = var01
MED.VALUE
y = var02
CRIMERATE
z = var03
%BIGLOTS
NOXCONC. /RANKS
LOW
HIGH
369

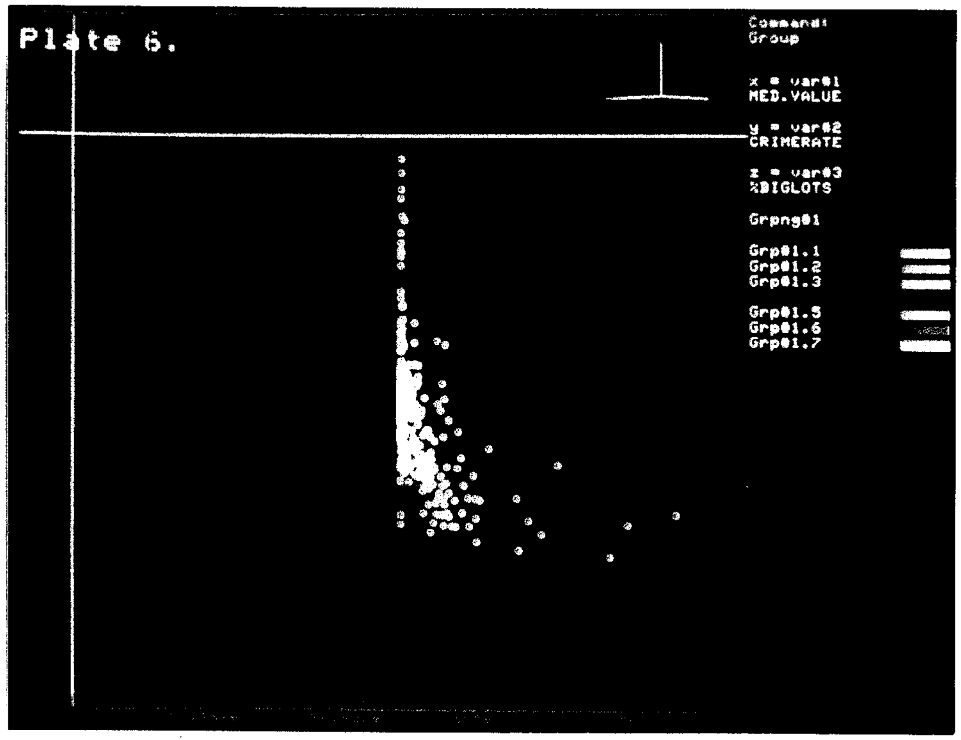


Plate 6.
Command:
Group
x = var#1
MED.VALUE
y = var#2
CRIMERATE
z = var#3
%BIGLOTS
Grpng#1
Grp#1.1
Grp#1.2
Grp#1.3
Grp#1.5
Grp#1.6
Grp#1.7

B

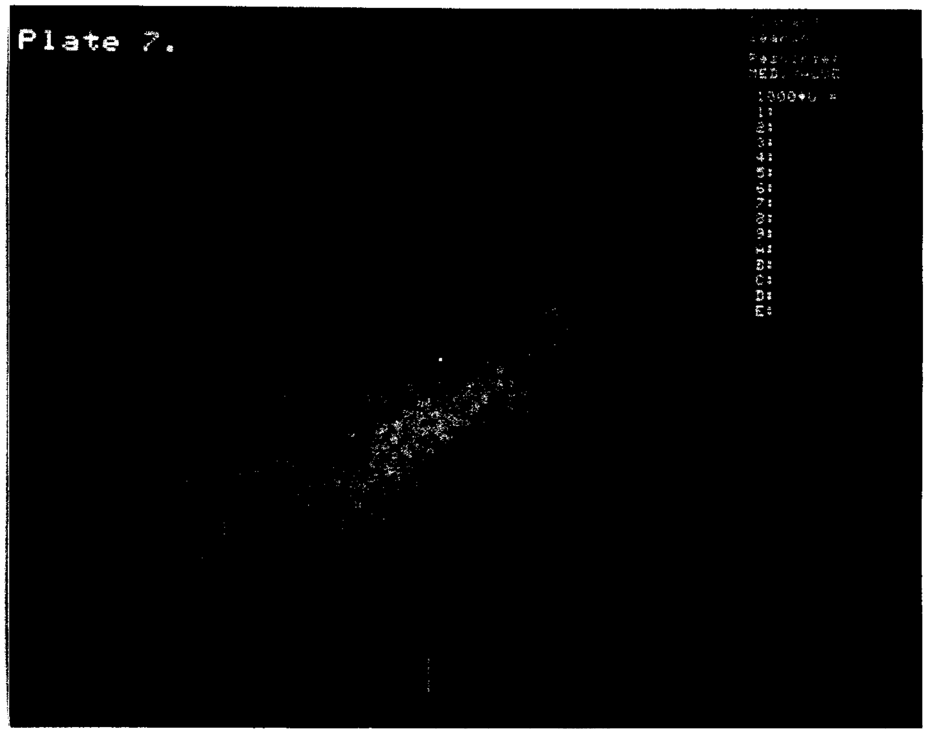


Plate 7.
MED.VALUE
1000*U =
1:
2:
3:
4:
5:
6:
7:
8:
9:
A:
B:
C:
D:
E:

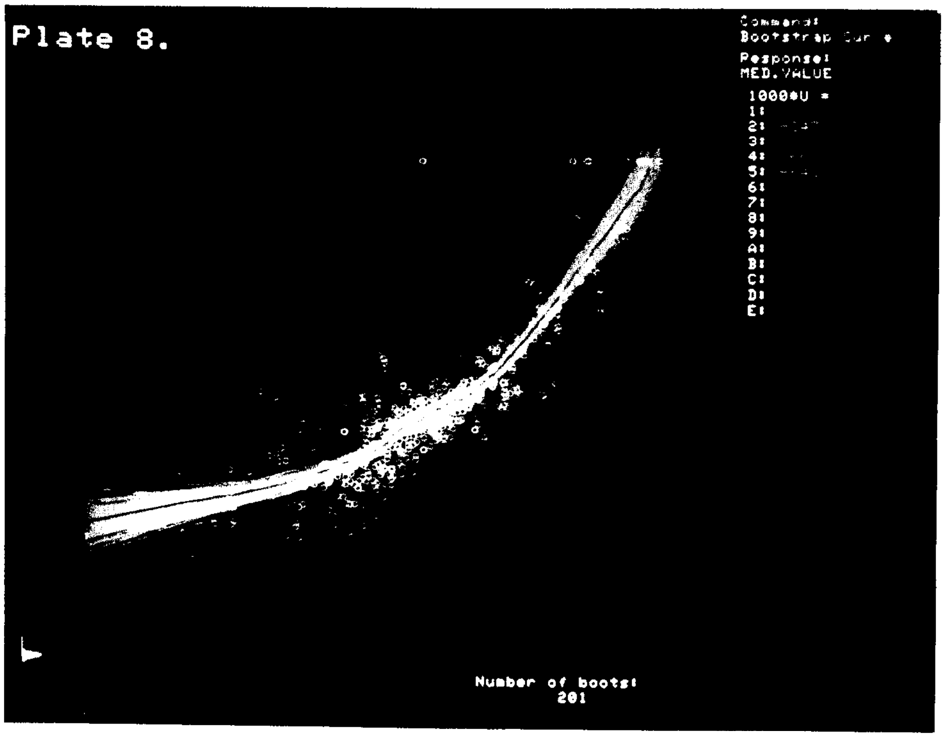


Plate 8.
Command:
Bootstrap Cur e
Response:
MED.VALUE
1000*U =
1:
2:
3:
4:
5:
6:
7:
8:
9:
A:
B:
C:
D:
E:
Number of boots:
201

C

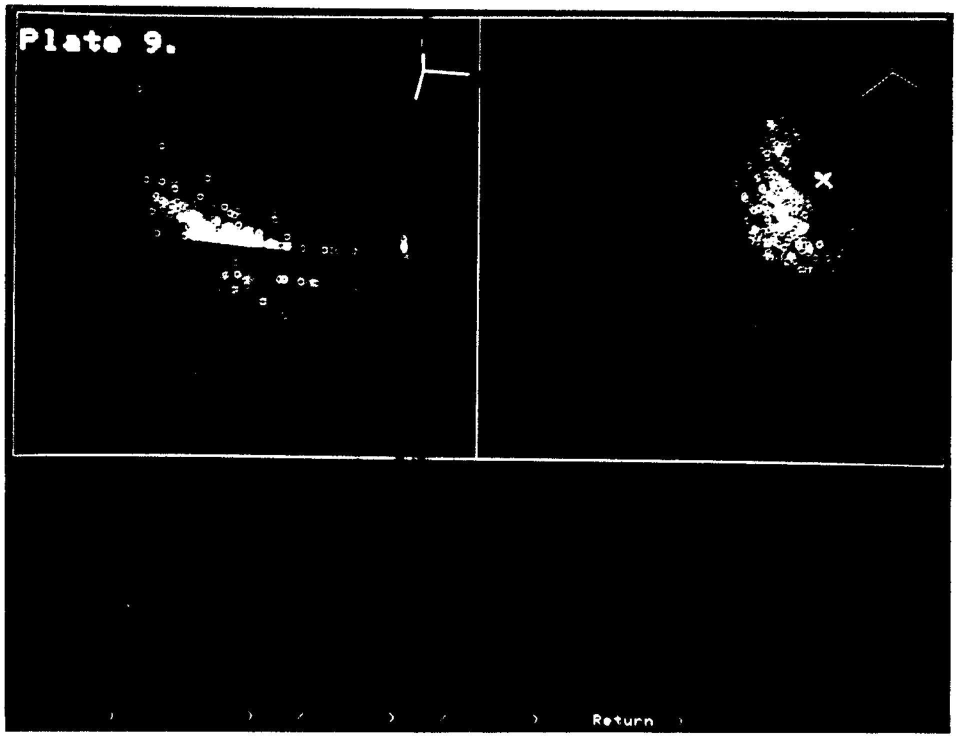


Plate 9.
Return

Chapter 1

INTRODUCTION

This paper reports on work in a new branch of research in statistical methods: applications of interactive computer graphics. The concrete result of this research is several programs written for the Orion I workstation, an experimental computer graphics system built at the Stanford Linear Accelerator Center in 1980-81. These programs are examples of some new methods for data analysis made possible by advances in microprocessor and computer graphics technology. These methods can be used on relatively inexpensive and soon to be widely available machinery (comparable to the next generation of personal computers).

I discuss my programs and the machinery in the Orion I workstation (which is not my work) in some detail. These details are not important in themselves, but they should illustrate how interactive graphics can contribute to statistics. The collection of hardware and software in Orion I is not a complete or definitive answer to anything. It is an example that, if successful, should suggest directions for future work.

In the course of this paper, it should become clear that there are differences between research on graphical methods and other, more mathematical branches of statistical research. Because the goal of research in graphical methods is to invent new ways of looking at data, the problems to be solved are more primitive than those addressed in some other parts of research in statistics.

In Chapter II, I suggest some general principles for applications of interactive graphics in statistics.

In Chapter III, I describe the Orion I workstation. Though I have made no significant contribution to the design or construction of the hardware in Orion I, I discuss it in some detail. I do this because, in this branch of statistical research, unlike most others, the physical machinery has a critical influence on what methods are possible or impossible, on what methods work well or poorly.

Chapter IV describes a "user model" for my programs and some basic operations for exploring many-dimensional data.

Chapter V discusses interactive projection pursuit methods, specifically projection pursuit regression.

Chapter VI describes a method for looking at many-dimensional data called 'Multiple Views'.

Chapter VII lists a number of areas for future research.

Because it is impossible to do justice to an interactive graphics system with a verbal description, we are making two films, which demonstrate some parts of the programs discussed below. The first film, called "Exploring Data with the Orion I Workstation", shows how some of the more basic operations can be used to do clustering. It corresponds very roughly to chapters IV and VI. The second film, called "Projection Pursuit Regression with the Orion I Workstation", corresponds to chapter V.

Chapter 2

GENERAL PRINCIPLES

In this chapter I relate some principles about applications of computer graphics to statistics. Because this is a fairly new field of statistical research, I feel it is worthwhile to discuss it in general terms, before getting to the details of my own work. To begin with, I will describe the kind of statistical problems that computer graphics can help solve. Then I give some examples of graphical methods. The first are simple examples for one-dimensional data; then I outline three approaches to many-dimensional data.

2.1 DESCRIPTION

Using graphical methods in statistics means looking at pictures of data. So it is no surprise that the purpose of graphical methods is primarily descriptive. Graphical methods are most useful for illuminating features of a particular data set; they are less useful for making formal generalizations. Graphical methods are subjective; the results of a graphical method depend on an individual's perception and interpretation.

The basic goal of statistical description is to "look at the data and see what is going on". Description has two parts:

1. Exploration

   Good description should reveal any interesting features of a data set. We especially want to be able to discover unanticipated kinds of structure in data.

2. Summary

   We need to report to others what we discover through exploration. It is often important to have concise summaries of particular aspects of data. It is also often important for summaries to be objective; in contrast, useful techniques for exploration tend to be subjective.

Since the time of R.A. Fisher, most statistical research has concentrated on inference. An inference is a generalization from a given data set to some larger population (real or hypothetical) from which the data set is presumed to be a sample. Statistical inference makes generalizations based on a model. Models used in inference usually have a deterministic part, that models structure in the population, and a probabilistic part, that models the way the data set is sampled from the larger population.

Inference is most powerful at answering specific questions about the relationship of a data set to a particular model. If prior knowledge about the data is limited, it may be difficult to construct a reasonable model--or to choose the right questions to ask.

Description solves more primitive, and fundamental problems than inference. Exploration and summary may be ends in themselves; some other goals are:

1. Informal Generalization.

   Description concentrates on features of the particular data set at hand; we do not draw (formal) conclusions about larger populations. Sometimes informal generalizations, based on common sense, are either good enough or the best we can hope to do.

2. Constructing models.

   Inference relies on models; good modeling requires looking at data.

3. Checking models.

   Even when we think we have sufficient prior knowledge about our data for inference to be appropriate, it is necessary to check the data for deviations from our assumptions. It is especially important to be able to detect unexpected kinds of deviations.

Methods for description can be sorted out into numerical methods, passive graphical methods, and interactive graphical methods. To illustrate the nature of these methods, I take a simple example from Efron [18]:

Consider two sets of one-dimensional data: A = { 94, 197, 16, 38, 99, 141, 23 } and B = { 52, 104, 146, 10, 50, 31, 40, 27, 46 }. These numbers are counts from an experiment on mice; set A is the treatment group; set B is the control group. I next consider ways to compare these two groups of numbers.

A traditional approach would be to model each set as a collection of independent, identically distributed observations from some underlying distribution. Comparing the two sets of numbers would be done by making inferences about the two underlying distributions. For example, if the two distributions are assumed to be Gaussian, a two-sample t-test would be an appropriate comparison.

With the limited information presented here, we cannot chosse reasonable distributions to model the two samples. In fact, we cannot even assume that the numbers represent <u>independent</u> observations.

However, we can still ask for comparisons and other descriptions of the two sets of numbers. For example, we might be using this small data set to help construct a model for inference in later experiments.

2.1.1 Numerical methods

Numerical methods are particularly appropriate as concise and objective summaries of particular aspects of data. The quantities used are often closely related to methods for inference, so numerical methods have a natural connection to both formal and informal generalizations. Numerical descriptions are also useful for detecting expected types of deviation from a model.

A traditional way of comparing the two sets of numbers is: Compute the mean and the standard deviation for each. The mean and the standard deviation are numerical summaries of the location and the spread of each set of data. The mean of set A is 87; the standard deviation is 62. The mean of set B is 56; the standard deviation is 40. So set A seems to be larger and more spread out than set B.

Arguments for this recipe appeal to notions of optimality when the numbers in the data sets are realizations of independent, identically distributed Gaussian random variables. These arguments are unconvincing when i.i.d. Gaussian is not a reasonable model.

An alternative recipe takes the median as a typical value for each sample and the median absolute deviation from the median (MAD) as a measure of variability. The median of set A is 94; the MAD is 56. The median of set B is 46; the MAD is 15. Again, A seems larger and more spread out than B. This recipe has advantages when we expect certain kinds of deviation from the Gaussian model.

For a particular probability model and a particular notion of optimality there will usually be a best summary. No summary will be best for all models and all notions of optimality, though some may do well enough in a large number of cases.

For the purposes of description, the performance of summary under any particular model is not the most important issue. If we are interested in illuminating features of a single data set, then notions of optimality based on probabilty models for the origin of the data may not be so important. The two recipes mentioned above are very useful, even when we know that the appropriate underlying models do not hold. The strongest advantage that the mean or the median have over more "robust" alternatives is that they are very simple. We can understand what the mean and the median tell us about the data easily. It is often difficult to understand more complicated, though more "robust" and "efficient" alternatives.

2.1.2 A graphical method: the histogram

An old graphical tool for description of univariate data is the histogram [7]. The histogram is an excellent tool for both exploration and summary. Looking at a histogram, we easily see many features of the data set, such as skewness, multiple modes, or outliers. For any single feature, there is usually be good numerical summary, such as the third moment of the sample for skewness. However, many numerical summaries would be required to capture some small fraction of the information present in the histogram. In addition, the histogram captures the "big picture" in a way that cannot be reproduced by any number of numerical summaries.

Looking at the two histograms in fig. 1 tells us the same thing that the numerical descriptions did: set A seems larger and more spread out than B. We also immediately see several other things that were not captured by the numerical summaries. Both sets of numbers are skewed (which is not surprising, since they are counts). Also set A seems to be bimodal.

Of course, with data sets of size 7 and 9, we do not take these indications of shape very seriously. If we had, say, 70 and 90 observations instead, it might be worth trying to interpret and understand the indication of bimodality in set A, the treatment group.

Numerical and graphical methods are not competitive; they are complementary. Graphical methods are not tied closely to any assumptions

```
   |
AAA|BBBB
   |BBB
 AA|B
  A|B
  A|
   |
```

Figure 1: Comparing two samples with histograms

or underlying model of the data. They expose data to human perception, which makes it possible to detect unexpected kinds of patterns. Interpretation of a graphical description is subjective. In contrast, numerical descriptions can be very efficient at summarizing the relationship of data to a model, when that model is appropriate. Numerical methods are objective; the mean is the mean for anyone.

The distinction between subjective and objective methods for data analysis is important. Each has its place. In scientific research, subjective methods are essential for discovering and interpreting unexpected features of data. On the other hand, if, for example, the results of analysis are used to determine public policy, it is necessary to have methods which are free from personal bias and can be agreed to by individuals with conflicting interests.

2.1.3 Interactive graphics

Our research with Orion I emphasizes interaction and real-time motion graphics. These are particularly useful for the exploratory part of description. We are developing methods for discovery, understanding, and summary of "structure" in many-dimensional data. Since, in general, we have little reliable prior knowledge about our data, we are especially interested in methods that allow us to discover unanticipated kinds of structure in data.

This is where interactive graphics has much to contribute. First of all, computer graphics can quickly expose many different views of data to human perception, so that the data analyst can detect many kinds of patterns in the data. The analyst can interpret apparent patterns using his knowledge of the context of the problem in which the data arose. Also, the analyst can select directions for further exploration using results obtained so far. Problems that are extremely difficult to solve by automatic and objective methods often become trivial with interactive methods -- because of the addition of human intelligence. Interactive graphics combines human talents for perception of patterns and judgement using the full context of a problem with a machine's ability to do rapid and accurate computation.

I next give some examples of how interactive graphics can be applied to one-dimensional data, using the histogram. Unfortunately, the histogram is such a good tool that the methods I suggest are, at best, only slight improvements. However, they should indicate the general nature of interaction.

2.1.3.1 Variable bin size

To draw a conventional histogram, we must choose a bin size. This is usually done by trial and error. On an interactive graphics machine we can provide a dial whose setting determines the bin size. Then we can choose the bin size by turning the dial and watching the histogram change as the bin size changes. The presence of the dial makes it easy to choose a pleasing value of the bin size. More importantly, watching the histogram change in a continuous way as we turn the dial lets us see aspects of the data set that are not captured by any single histogram.

2.1.3.2 Re-expression

An issue that arises in description of univariate data is <u>re-expression</u> [9,46,42]. Put very simply, we may want to look at histograms of simple functions, or re-expressions, of the original data. An interactive graphics system can help us choose a re-expression.

A convenient and widely discussed class of re-expressions is by powers. That is, if our original data set is $\{x_i\}$, then we look at a histogram of $\{x_i^{\alpha}\}$ for some real number, α. (It is standard practice to replace x_i^{α} by $\log(x_i)$ when $\alpha = 0$.)

On a graphics machine we can provide a dial that controls the value of α. We can then watch the histogram change in a continuous way as we turn the dial and change the value of α. As with the variable bin size, this not only provides a way of choosing a value of α, but also shows us more about the data than any single, static view.

2.2 GRAPHICAL METHODS FOR EUCLIDEAN DATA

We look at univariate data with histograms.

We look at two-dimensional data with conventional scatterplots. In conventional ~~scatt~~scatterplots, observations are represented by points, which are plotted at horizontal and vertical positions corresponding to the values of two variables.

To look at three-dimensional data, we draw a three-dimensional version of the scatterplot. Real-time motion graphics makes it possible to draw pictures that appear three-dimensional. We subject the data to repeated small rotations and display the projection of the rotated data as points on the two-dimensional screen. If we can compute and display rotations fast enough (>10 per second) then we get an illusion of continous motion. Apparent parallax in the motion of the points lets us see the point cloud as a three-dimensional object.

There is no completely satisfactory method that lets us look at more than three variables at a time. Three basic approaches to viewing many-dimensional structure are being studied with Orion I. They are:

1. Higher-dimensional views

2. Projection Pursuit

3. Multiple Views

2.2.1 Higher-dimensional views

The idea here is to represent as many variables as possible in a single picture.

A simple way to add dimension to a picture is to use color. We start with a three-dimensional scatterplot. We add a fourth variable to the picture by giving each point in the scatterplot a color that depends on the value of a fourth variable. With an appropriately chosen color spectrum, we easily see simple relationships between the fourth variable and position in the three-dimensional space. Our ability to perceive distinctions in color is not as precise as our ability to perceive position in space; we should expect to miss subtle or complicated relationships between a color variable and three position variables. Color works best for a discrete variable that has on a small number of possible values.

Other tricks let us increase the dimension of a picture. For example, we can represent each observation in the scatterplot by a circle, rather than by a simple point. Then the radius of the circle can represent the value of a additional variable.

In a simple scatterplot, observations are represented by featureless points. We add dimension to the picture by replacing points with objects that have features, such as color, size, and shape. These features represent variables in addition to those represented by a point's position in the scatterplot. These "featurefull" objects are sometimes called <u>glyphs</u> [32,47].

With each additional dimension, the picture becomes more complicated and difficult to interpret. We need experience to determine successful ways of adding dimension and to understand the limitations of each method.

2.2.2 Projection Pursuit

The basic problem with adding dimension to a single picture is that the picture quickly becomes impossible to understand. The alternative is to restrict our picture to a few (<= 3) dimensions and then select low-dimensional pictures that capture interesting aspects of the multivariate structure in our data. This is the basic idea of projection pursuit.

Projection pursuit is discussed in detail in chapter V.

2.2.3 Multiple Views

Usually no single low-dimensional view will capture everything that is interesting about a many-dimensional data set. Therefore we will need to look at several views of the data. It may be possible to understand more about the many-dimensional structure in the data if we look at several views simultaneously and connect the contents of the views in some way.

For example, draw two two-dimensional scatterplots side by side. Then connect points corresponding to the same observation by drawing them in the same color. With a good choice of a coloring scheme, we can see how structure in one scatterplot maps into the other scatterplot. The coloring scheme is best determined interactively.

Multiple views are discussed in more detail in chapter VI.

2.3 STRUCTURE AND SHAPE

The principal goal of the methods discussed above to discover and understand structure in many-dimensional data. What we mean by structure is any apparent pattern or interesting feature in a graphical (or numerical) description of data. Structure is a subjective perception. Quantitative measures of specific aspects of structure can be very useful, but a small number of quantitative measures cannot capture all important aspects of structure. It is also very difficult to choose quantitative measures of structure that respond appropriately in unexpected circumstances.

Suppose all the variables in our data set are reasonably represented by real numbers. We call this type of data euclidean. This type of data characterizes classical multivariate analysis. For euclidean data, structure usually refers to the "shape" of the data set in the many-dimensional data space.

In classical multivariate analysis, data is usually summarized by the mean vector and the covariance matrix. The mean vector and covariance matrix completely describe the position, orientation, and eccentricity of ellipsoidal shapes. However, they cannot describe or detect any more complicated shapes. In our experience with Orion I, we have seen no data sets whose shape is well described by ellipsoids. In fact, a data set whose shape can be accurately summarized by a mean vector and a covariance matrix is one that shows little interesting structure.

Two simple examples of non-ellipsoidal shape are: 1) the observations are clustered and 2) the observations lie close to a lower-dimensional surface or manifold. It is, in addition, easy to imagine non-ellipsoidal shapes that cannot be classified as either clusters or manifolds.

2.3.1 An example

The following simple type of structure is illustrated in our films: "Exploring data with Orion I" and "Interactive projection pursuit regression with Orion I".

A typical thing that we would like to know about a data set is the following: Does the data set separate in a natural way into distinct groups? (This is not exactly the same thing as clustering.)

In our films, we look at a data set presented by D. Harrison and D.L. Rubinfeld [35,6]. They measured 14 variables for each of 506 census tracts in the Boston Standard Metropolitan Statistical Area. Harrison and Rubinfeld were concerned with the dependence of housing value (represented by the median value of owner-occupied houses) on air pollution (represented by nitrogen oxide concentration). The remaining 12 variables measured other quantities thought to influence housing value, such as crime rate, average number of rooms, etc.

In the exploration film, we partition the data set into a few natural subsets, based on observed structure. In our second film, on projection pursuit regression, we look at the dependence of housing value on the other thirteen variables. The regression film builds on the results of the explore film.

To study the dependence of housing value, we fit a model that predicts housing value as a function of the thirteen predictors. Before fitting any model, we must consider whether it is appropriate to fit one model for all the data. If the data set separated in a natural way into distinct and internally homogeneous groups then we would want to consider fitting different regression models for each group.

Most clustering algorithms rely on a notion of distance in the data space to partition a data set into isolated clumps. Observations are considered similar if they are close together and dissimilar if they are far apart. A cluster is a group of points that are close to each other and far from any other points.

With Orion I, we can use other criteria besides separation in a distance measure to partition a data set. In particular, we can use subjective perception of patterns to define natural groupings. A data set may divide into two groups, which follow clearly distinct patterns. Yet the difference between the groups may not be easily summarized by a measure of distance.

In the explore film, we show how the Harrison-Rubinfeld data can be divided into several groups. The major division turns out to be between urban and suburban-rural census tracts. However, the urban and suburban tracts do not form isolated clumps. Instead, in certain views, they lie close to two intersecting, perpendicular planes. Because the planes intersect, the groups are not isolated in most distance measures. But the separation in the two groups is obvious when seen.

Chapter 3

ORION I

In this chapter I describe the Orion I system. I describe the hardware and aspects of programming the system in detail. I do this for two reasons:— first, to give a concrete reference for the applications of the system described in the following chapters and second, to give an indication of the skills that are required to do research in new graphical methods for statistics. This chapter is based largely on Friedman and Stuetzle [26] which is a detailed description of the hardware and the decisions taken in its design.

3.1 REQUIREMENTS FOR MOTION GRAPHICS

Real time motion graphics requires hardware with the ability to compute and draw new pictures fast enough to give the illusion of continuous motion. Five pictures per second is a barely acceptable rate. Ten to thirty times per second gives smoother motion and more natural response for interaction with a user.

Orion I uses real time motion to display three-dimensional scatterplots. We can view three-dimensional objects on a two-dimensional display by continuously rotating the objects in the three-dimensional space and displaying the moving projection of the object onto the screen. In a scatterplot, the object we want to look at is a cloud of points. A typical point cloud will contain from 100 to 1000 points. So our hardware must be able to execute the viewing transformation, which is basically a multiplication by a 3x3 rotation matrix, on up to 1000 3-vectors ten times per second. The system must also be able to erase and draw 1000 points ten times per second.

3.2 HISTORY

Orion I is the youngest descendant of a graphics system called Prim-9, which was built at SLAC in 1972 [21]. Prim-9 was used to explore up to nine-dimensional data. It used real time motion to display three-dimensional scatterplots. Through a combination of picturing and rotation, a user of Prim-9 could view an arbitrary three-dimensional subspace of the 9-dimensional data. Isolation and masking were used to divide a data set into subsets.

The computing for Prim-9 was done in a large mainframe computer (an IBM 360/91) and used a significant part of the mainframe's capacity. A Varian minicomputer was kept busy transferring data to an IDIIOM vector drawing display. The whole system, including the 360/91, cost millions of dollars. The part devoted exclusively to graphics cost several hundreds of thousands of dollars in 1965.

Successors to Prim-9 were built at the Swiss Federal Institute of Technology in 1978 (Prim-S) and at Harvard in 1979-80 (Prim-H) [16].

Prim-S used a DEC-10, a PDP-11/34, and an Evans and Sutherland Picture System 2 as analogs of the IBM 360/91, the Varian, and the IDIIOM.

Prim-H is based on a VAX 11/780 computer and an Evans and Sutherland Picture System 2. It incorporates a flexible statistical package (ISP). The system costs several hundreds of thousands of dollars. Computation for rotation is done by hardware in the Evans and Sutherland. The VAX is shared with perhaps two dozen other users and has limited capacity for intensive real time computation.

3.3 ORION I HARDWARE

The Orion I workstation was designed and built in 1980-81 by Jerry Friedman and Werner Stuetzle with help from the members of the Computation Research Group at the Stanford Linear Accelerator Center.

There are two ways in which Orion I is a substantial improvement over previous Prim systems: price and computing power. The total cost for hardware in Orion I in 1981-82 is less than $60,000.[1] The computing power in Orion I is equivalent to that of a large mainframe computer (say one half of an IBM 370/168 or three times the VAX 11/780 used in Prim-H) and is devoted to a single user.[2] The hardware and important considerations in its design are described in detail by Friedman and Stuetzle in [26]. The important parts of Orion I are:

1. the master processor.

2. the graphics device.

3. the arithmetic processor.

4. the input-device(s).

[1] A preliminary version of Orion I, with a lower resolution graphics device, cost less than $30,000. Black and white graphics systems, with all the capabilities of the earlier Prim systems, can be bought for about $8,000 [4].

[2] For comparison, it was common about five years ago for all of the computing in a typical university computer center to be done on a single 370/168.

```
*---* *----------------------------------* 2400 *----------*
|   |=|                                  | baud |          |
|   |=| Master processor:                |<---->| terminal |
| M |=| SUN microcomputer                |      |          |
| U |=| (Motorola 68000 microprocessor,  |      *----------*
| L |=|  256K RAM, 2 serial I/O ports)   | 2400 baud to host
| T |=|                                  |<------------------------
| I | *----------------------------------*                         |
| B |                                                *---------*  |
| U | *---* *------------------------------------*   | output: |  |
| S | | i | |                                    |-R->| 19 inch |  |
|   |=| n |=| Graphics Device:                   |-G->| color   |  |
|   |=| t |=| Lexidata 3400 raster scan display  |-B->| monitor |  |
|   |=| e |=| (1280x1024x8 frame buffer,         |    |         |  |
|   |=| r |=|  display processor,                |    *---------*  |
|   |=| f |=|  3 256x8 color look-up tables)     |                 |
|   |=| a |=|                                    |    *--------*   |
|   |=| c |=|                                    |<---| input: |   |
|   | | e | |                                    |    |tracker-|   |
|   | *---* *------------------------------------*    |ball    |   |
|   |                                                 *--------*   |
|   | *---*                                                        |
|   | | i | *------------------------------------*                 |
|   |=| n |=|                                    |                 |
|   |=| t |=| Arithmetic Processor:              |                 |
|   |=| e |=| 168/E                              |                 |
|   |=| r |=| (approx. 1/2 370/168 cpu,          |                 |
|   |=| f |=|  96K bytes data memory,            |                 |
|   |=| a |=|  48K bytes program memory)         |                 |
|   |=| c |=|                                    |                 |
|   | | e | *------------------------------------*                 |
|   | *---*                                                        |
|   |                                                              |
|   | *---------------------*         *-----------------------*   |
|   | |                     |         |                       |   |
|   |=| Fast Link Interface | coaxial | Host Computer:        |<-
|   |=|                     |  cable  |                       |
|   |=| (emulates IBM 3270  |<--------|  IBM 3081             |
|   |=|  terminal,          |         |                       |
|   |=|  approx. 1M baud    |         |  (used for preparing  |
|   |=|  data transfer)     |         |   programs and data,  |
|   | |                     |         |   not for computation)|
|   | *---------------------*         |                       |
|   |                                 *-----------------------*
*---*
```

Figure 2: A diagram of the Orion I workstation

3.3.1 The master processor

The master processor controls the action of the other parts of the system (except the host) and handles the interaction with the user.

The master processor is a SUN microcomputer, based on the Motorola 68000 microprocessor. The SUN is a single (MULTIBUS) board computer developed by the Stanford Computer Science Department for the Stanford University Network. The SUN has 256k bytes of RAM (random access memory). It has two RS232 serial IO ports that connect to a terminal and to a host computer (slow link). It communicates with the other parts of the system through 16 bit parallel interfaces, designed and built by Werner Stuetzle, that plug into the MULTIBUS.

The SUN is programmed mostly in Pascal. A few critical routines for picture drawing are written in MC68000 assembly language and some system programs (monitor/debugger) are written in C.

The SUN board costs about $3,500.

3.3.2 The graphics device

The graphics device is a Lexidata 3400 raster scan display. The Lexidata contains:

1. A frame buffer

   The frame buffer is memory that stores the current picture [22,43]. It has eight bits of memory (one byte) for each pixel in a raster of 1280x1024. Each pixel in memory corresponds to a dot on the screen; so the screen has a resolution of 1280x1024.

2. A display processor

   The Lexidata 3400 contains a display processor that is used for drawing vectors, circles, and characters. The 3400 display processor can be (micro)programmed in its own (microcode) assembly language. So far, we use the IDOS system of graphics routines provided by Lexidata.

3. Color look up tables

The byte of memory for each pixel determines its color, indirectly, through a look-up table. There are three look-up tables, one each for the red, green, and blue guns in the monitor. Each look-up table has slots with addresses from 0 to 255. Each slot contains a value from 0 to 255 that determines the intensity of the corresponding gun. The color of a pixel is determined by the settings contained in entries in the look-up tables corresponding to the value of its byte. Thus, at any time, there may be 256 different colors on the screen, from a potential palette of 2^{24}.

4. A monitor

The pictures stored in the frame buffer are displayed on a 19 inch color monitor. A monitor for the high resolution frame buffer costs about $6,000 and for the low resolution (640x480) frame buffer about $2,500 (in June 1982).

Color raster graphics devices like the Lexidata are cheaper and more flexible than the black and white line drawing displays used in earlier Prim systems. For more details on the differences between line drawing and raster displays see Foley and VanDam [22].

When purchased in January 1982, the 1280x1024 pixel frame buffer and display processor cost about $40,000. An earlier version of Orion used a 640x480 frame buffer, which provides more than adequate resolution for statistical applications and costs about $10,000. Memory and, therefore, frame buffer prices are falling rapidly. Recently (June 1982), a raster scan device with a 1024x780 frame buffer and a faster display processor than the Lexidata has been announced that costs about $15,000.

3.3.3 The arithmetic processor

The arithmetic processor executes demanding numerical computations rapidly.

In an early version of Orion I, all computations were done by the master processor, the SUN computer. This system had all the capabilities of earlier Prim systems. In particular, the MC68000 microprocessor has sufficient computing power to rotate 1000 points smoothly.

In the current system, we use a processing unit called a 168/E [39,40]. The 168/E was developed by SLAC engineers for the processing of particle physics data and has about half the speed of the true 370/168. It emulates an IBM 370/168 central processing unit without channels and interrupt capabilities. Because it has no input/output facilities, it is strictly a slave processor; its action is controlled by the master SUN computer. Our version of the 168/E has 96k bytes of data memory and 48k bytes of program memory.

The 168/E can execute our most demanding real time computation, a sophisticated smoothing algorithm, on large data sets (more than 500 observations), more than ten times a second. This smoothing algorithm [27] is much more demanding than simple rotation.

The 168/E is programmed in FORTRAN. The FORTRAN programs are compiled with standard IBM FORTRAN compilers on an IBM 3081 host computer. The object code is then translated into 168/E microcode.

The 168/E costs about $5,000 (in June 1982). Unfortunately, it is not commercially available.

3.3.4 The input device(s).

My programs are controlled through a device called a trackerball [22,43], which is a hard plastic ball about 3 inches in diameter set into a metal box so that the top of the ball sticks out. The ball can be easily rotated by hand. The position of the ball determines the values of two coordinates. The two coordinates are used for input quantities that need to be varied continuously. For example, the trackerball often determines the angles of a rotation; the apparent motion of an object on the screen mimics the motion of the trackerball under a user's hand. The trackerball also has six switches, which are used for discrete input to programs.

The Lexidata trackerball costs about $2,000. Less precise trackerballs are available for less than $1,000.

3.4 PROGRAMMING ORION I

There are, at last count, five processors in the Orion I system: the master processor, the arithmetic processor, the display processor in the graphics device, a microprocesser in the trackerball, and a microprocessor in a high speed (≃ 1 megabaud) serial interface between Orion I and the IBM 3081 host computer. Each of these processors has its own programs. To develop new methods on Orion I, it is necessary to program the host computer as well.

Orion I makes use of programs written in 14 languages: The SUN computer is programmed in C, Motorola Pascal, and Motorola 68000 assembly language. The arithmetic processor, the 168/E, is programmed FORTRAN, MORTRAN, IBM 370 assembly language, and its own microcode. The display processor is programmed in its own (microcode) assembly language. The microprocessors in the trackerball and the high speed serial interface are programmed in their own assembly languages. The host computer is programmed in IBM Pascal/VS, MORTRAN, FORTRAN, and assembly language.

The programs I describe in the following chapters are written in Motorola Pascal and Motorola 68000 assembly language for the master processor, the SUN, and in MORTRAN and FORTRAN for the arithmetic processor, the 168/E. These programs are edited and then cross-compiled or cross-assembled on the host,[3] the IBM 3081. Data sets are also prepared on the 3081. Programs and data are downloaded to the SUN computer and the 168/E through the high speed serial interface that connects the MULTIBUS and the 3081.

---

[3] A host is not an essential part of the system. The SUN computer could be made completely independent of the host for about $10,000 in peripherals, such as a disk drive, printer, etc.

Chapter 4

ELEMENTARY APPLICATIONS

The programs described in this chapter reproduce most of the functions of the Prim-9 system [21], with the addition of some elementary uses of color.

In the next few sections, I begin to describe a _user model_ for some of the programs written for Orion I that are described in this and the following chapters. "User model" [43,22] refers to a set of abstract concepts that let a user think about what a program is doing, without having to understand details internal to the program. The user model defines the kinds of _objects_ that a program acts on and the _actions_ or _commands_ that affect objects.

4.1 _USING ORION I_

4.1.1 _Input_

Foley and VanDam [22, p. 183] distinguish five classes of logical input devices. In our system, the trackerball serves as a _locator_, which indicates position and/or orientation, and a _valuator_, which chooses a single value in the space of real numbers. The switches on the trackerball serve as _buttons_, which select from a set of possible aternatives. The ball and a particular switch, called the enter switch, combine to be a _pick_ device, which selects a entity displayed on the screen. Our system also includes a _keyboard_, the fifth logical input device, which is used to input a character string. My programs do not use the keyboard.

A user of my programs executes commands either by using a switch (button) that is identified with a particular command, or by picking an item from a menu of choices listed on the screen.

4.1.2 Output

In my programs, the screen is divided into three regions (see Plate 1). A strip on the bottom of the screen, one character high, shows six labels that indicate the current functions of the six switches on the trackerball. A region approximately 16 characters or 200 pixels wide on the right side of the screen is reserved for the display of assorted information about the current state of the program. The remaining area, approximately 1000x1000 pixels, is used for the display of pictures of data. Occasionally, the pictures of data will be temporarily overwritten with a menu of commands or of options for a particular command.

4.2 DATA

A data set is a collection of pieces of information, represented by numbers. Simple data sets are structured as a two-dimensional arrays of observations versus variables.

In these programs, I distinguish two types of variables: euclidean and categorical.

Euclidean variables take on continuous, ordered values which can be reasonably represented by real numbers. Examples are variables such height, weight, blood pressure, etc.

Categorical variables take on discrete values, called categories. For example, the variable, "make of car", might have categories: Ford, Chevrolet, Chrysler, Toyota, Datsun, Mercedes. In the programs described below, categorical variables are arbitrarily restricted to have no more than seven categories. Usually the values of a categorical variable are assumed to have no natural ordering. It will be convenient for us to be able to treat ordered, discrete variables as either euclidean or categorical.

The explicit consideration of non-euclidean data is a conscious extension of previous Prim systems. There are, at present, few good graphical tools for exploring categorical and other types of non-euclidean data. This is an important area for research. Real data sets often include many different types of data; we need graphical tools that let us look at structure involving combinations of euclidean, categorical, and other non-euclidean types of data.

Suppose a data set contains p_e euclidean variables and p_c categorical variables. The p_e euclidean variables may be thought of as a the canonical orthogonal basis of a p_e-dimensional real inner product space [34], called the euclidean data space. The p_c categorical variables form the categorical data space, which is not a familiar mathematical object.

We call the cartesian product of the p_e-dimensional euclidean data space and p_c-dimensional categorical data space the data space. An observation is an element of the data space. A data set is simply a set of observations.

4.3 THREE-DIMENSIONAL SCATTERPLOTS

Motion graphics makes it possible to draw a three-dimensional analog of the two-dimensional scatterplot. In the following, scatterplots are assumed to be three-dimensional unless otherwise noted.

The picture we see in a three-dimensional scatterplot is drawn in the view space. The view space is spanned by three (basis) vectors: screen-x, screen-y, and screen-z. Screen-x is horizontal in the plane of the display screen; screen-y is vertical in the plane of the display screen; screen-z is perpendicular to the plane of the screen, pointing out.

There are two steps in mapping the data space to the view space. First, the data space is mapped onto a three-dimensional space called the world space, by a projection. Second, the world space is mapped onto the view space by the viewing transformation. Rapid, repeated applications of a slightly modified viewing transformation give an illusion of continuous motion; parallax in this motion lets us see three dimensions.

4.3.1 The projection

The projection can conceivably be any mapping from the many-dimensional data space to the three-dimensional world space. It could be a function of both euclidean and categorical variables. Some methods for data analysis may require non-linear mappings (such as "twists", in Tukey [48]). However, in the programs I am discussing below, the projections are restricted to orthogonal projections from the euclidean data space to the world space.

Two commands allow us to determine the projection onto the world space.

1. We can get a new projection.

   To do this, we choose three of the euclidean variables. The new projection maps the data onto their linear span.

   Mechanically, this is accomplished by using x, y, and z switches on the trackerball to step the index of the x, y, and z variables that determine the projection.

2. We can update the projection.

   We choose one euclidean variable by picking an item from a menu of the labels of the euclidean variables.

   The screen plane (screen-x vs. screen-y) corresponds at any moment to some two-dimensional linear subspace of the euclidean data space. The updated projection maps the euclidean data space onto the linear span of this plane and the basis vector corresponding to the chosen variable. This process can be thought of as throwing away the one-dimensional subspace of the world space corresponding to screen-z and replacing it by one of the euclidean variables.

   The update command is designed this way, which may seem unnatural at first, to let us mimic the search strategy of a Rosenbrock method for numerical optimization [44]. The update command is used for interactive projection pursuit, which is discussed in chapter V.

   Updating the projection allows us to see three-dimensional subspaces of the euclidean data space which are not simply the linear span of three variables. This method of updating does not allow us to select any three-dimensional subspace, but it does allow us to display any two-dimensional subspace on the screen. For the projection pursuit applications considered so far an arbitrary two-dimensional subspace is good enough.

4.3.2 The viewing transformation

In general, a viewing transformation is composed from translation, scaling, rotation, the perspective transformation, clipping, and hidden object elimination [22,43].

In the programs discussed here, only translation, scaling, and rotation are performed. Only the rotation can be controlled by the user. The translation and scaling are determined so that the point cloud fits on the screen, no matter what the angle of rotation.

The trackerball provides the angles of the rotation. If the trackerball is moved, the viewing transformation is changed and the points on the screen appear to rotate. The user also has the option of automatic rotation. Then the effect of the last motion of the trackerball is repeated, which gives smoother motion than can be achieved by hand.

Eventually, the full viewing transformation will be under user control.

Control of translation and scaling will permit the user to "zoom" and "pan", to examine parts of the point cloud more closely. Once we allow arbitrary translation and scaling it is necessary to "clip" points that would be drawn off the screen.

When the points are colored, or have other distinctive features, and the point cloud is dense, it is necessary to do hidden object elimination. Otherwise, there are disturbing artifacts in rotation. For example, if one side of the point cloud is red and the other side is blue, then the red points should obscure the blue points when the red side of the point cloud is in front. If we draw the picture in a naive way, we see the color of whichever points happen to be drawn last. For point clouds, hidden object elimination can be done simply by sorting the points by depth before drawing them on the screen.

When we are looking at clouds of featureless points, perspective does not seem to add much to the perception of three-dimensional structure. Also, some applications, such as projection pursuit, require the mapping of the data onto the two-dimensional space (screen-x,screen-y) to be orthogonal.

4.3.3 Coordinate Axes

We need to be able to tell which three-dimensional subspace of the euclidean data space we are looking at. We also need to see how the point cloud is oriented in that space. To satisfy these needs we draw, in a corner of the screen, an object called the coordinate axes or just the axes.[4]

---

[4] This object was called the dreibein (German for tripod) in previous Prim systems [21], and is sometimes refered to as the the gnomon in

If we are looking at a world space (see Plate 2), spanned by three variables, then we see the three coordinate axes as projected on the plane of the screen in the current rotation. As we rotate the data, the coordinate axes rotate also, but about a point in the corner rather than about the point in the center of the screen used by the point cloud. By comparing the motion of the point cloud to the motion of the axes we can determine the orientation of the point cloud in three dimensions.

When the world space is a more general three-dimensional subspace of the euclidean data space (see Plate 3), we will see a collection of vectors radiating from a common origin, each labeled with the index of the variable it represents. Each vector is the projection of a unit vector in the direction of the corresponding variable onto the world space.

In the case of the more general three-dimensional subspace, it is not easy to interpret the picture. These complex views arise and are most useful in the context of projection pursuit, discussed in the next chapter.

4.4 IDENTIFY

The Identify command (see Plate 4) lets us find out which observation is represented by a given point on the screen. The viewing transformation is frozen in Identify so that we see a two-dimensional picture. A cursor is positioned on the screen with the trackerball. The point on the screen nearest the cursor is highlighted by changing its color and drawing a circle around it. The index of the corresponding observation is then written on the screen near the point.

The highlighted point can also be temporarily deleted from the data set by moving the delete switch. Deleted points take no part in any actions and are unaffected by them, until restored. This feature is used, for example, in the regression program discussed in chapter V. Unusual observations can be deleted from the data set temporarily, so that they do not interfere with the construction of a regression model.

---

the computer graphics literature [22].

4.5 COLOR

As I have noted above, adding color is a simple way to increase the number of dimensions represented in a picture.

4.5.1 A color model

A color model is a parametrization of the set of colors that can be distinguished by the human eye. Color models are usually three-dimensional. Many different color models are used in computer graphics [22]; a convenient model for statistical applications is the HSV model, for Hue, Saturation, and Value. Hue denotes the distinction between red, yellow, green, cyan, blue, and magenta. Saturation refers to the purity of the color; a vivid red is highly saturated while a pale pink is not. Value measures the overall brightness. The HSV model represents colors as points in an inverted hexcone (a point-down "cone" that is hexagonal in cross section). The Hue coordinate is the angle of rotation about the axis of the cone. Saturation corresponds to the radial distance from the axis. The Value coordinate corresponds to height.

The point of the cone, at the bottom, corresponds to black. The center of the hexagonal top face of the cone is white. The six vertices of top face are the brightest and most saturated red, yellow, green, cyan, blue, magenta.

We use color to represent the values of an additional variable in a two- or three-dimensional scatterplot. All points in the scatterplot should appear equally bright, so when we choose a spectrum of colors to represent the values of a variable, we take colors with the same Value. Thus a variable is represented in color by varying Hue and/or Saturation. Varying Hue alone seems to work better than varying Saturation alone. Varying Hue and Saturation together may increase slightly the number of different colors that can be distinguished.

4.5.2 Representing categorical variables by color

We represent a categorical variable in a scatterplot by the Hue of the points -- a different Hue for each category. Thus, in the simplest picture, we see three euclidean variables, represented by the position and motion of the points of the screen, and one categorical variable, represented by the colors of the points. The categorical variable that determines the color of the points at any instant is called the color variable.

The Choose Color Variable command lets the user choose the color variable by selecting one of the categorical variables from a menu.

The user has the option of two coloring schemes. One scheme is designed for variables with unordered discrete values. Points take on one Hue out of seven: red, yellow, green, cyan, blue, magenta, or white. The other scheme is intended for ordered variables; points can have colors in a perceptually continuous range of Hues, from blue through magenta ~~to~~ red.

When we use a continuous range of Hues, three steps can be perceived easily. With carefully chosen Hues and some concentration by the user, from five to seven Hues can be distinguished. More than seven steps in Hue does not change the appearance of the picture. It is easier to perceive distinctions in Hue if we represent observations by larger objects than points, such as circles or crosses.

4.5.3 Representing euclidean variables

We represent euclidean variables by color using the Discretize command (see Plate 5). Discretize converts a euclidean variable into an ordered discrete variable. Because of our limited ability to perceive distinctions in a continuous range of Hues, the created discrete variable takes on values from 1 to 7. Two options are currently available:

1. Discretize by value.

   The observed range of the euclidean variable is divided into seven intervals of equal length. Each observation is given the value of the index of the interval into which it falls.

2. Discretize by ranks.

   The observed values of the euclidean variable are first replaced by their ranks, which are then divided into seven equal steps.

Another option for coloring euclidean variables is still to be added:

3. Boxplot coloring.

This method of coloring is based on Tukey's boxplots [46,42]. The basic idea is to represent the central values of a variable by a continuous range of colors as above. The negative and positive outliers would be given two Hues distinct from the range of Hues representing the center of the distribution.

4.5.4 Grouping

The Group command lets us interactively create or modify a categorical variable, based on the positions of the points on the screen (see Plate 6).

While executing the Group command, no rotation or other change in the viewing transformation is permitted, so we see a fixed two-dimensional scatterplot, which may show any two-dimensional subspace of the euclidean data space. The action of the Group command uses only the fixed positions of the points on the screen.

The Group command is used to change the values of one of the categorical variables. It can be used either to modify an existing categorical variable or to define a new categorical variable. In the latter case, a new categorical variable is added to the data set, all observations are initialized to category 1, and the values of the new variable are modified with the Group command.

The Group command is used to modify the values of a categorical variable by repeatedly moving sets of points into one of the (seven) categories of the variable. The destination category is chosen by using the trackerball and an "enter" switch to pick one of the categories from a list on the side of the screen. Each move is made by defining a rectangular region on the screen and moving all the points inside the rectangular region into the chosen category. The rectangular region is defined using the trackerball and the "enter" switch to position a cursor on the screen and to mark the rectangle's two corners.

The ability to modify existing categorical variables is useful because it lets us define a categorical variable based on several views of the data. We first find a view that shows structure that suggests a natural partition of the data set. This structure is summarized by creating a new categorical variable that records the partition. We can then project and rotate to a new view that suggests a refinement or other modification of the partition and modify the categorical variable accordingly. This process is illustrated in our film: "Exploring data with the Orion I workstation".

4.5.5 SelectCategory

We immitate the Isolation and Masking functions of Prim-9 using Group and a related command, called Select Category.

Select Category shows the user a menu of the categories of the current color variable. The user can then select which categories are to be active or inactive. Observations in inactive categories are invisible. They do not affect and are unaffected by any subsequent actions.

Chapter 5

INTERACTIVE PROJECTION PURSUIT

5.1 PROJECTION PURSUIT METHODS

The basic idea of projection pursuit is to choose low-dimensional views of a data set that capture aspects of its many-dimensional structure [38,21,30,27,28,25,36,37,10]. By temporarily reducing the dimension of the data, we can sometimes replace a problem that is difficult or impossible to solve with one or several "smaller" problems that are more manageable. This is especially true for graphics. It is hard to understand pictures that show many variables at once; it is easy to understand one-, two-, and three-dimensional pictures, that is, histograms and two- or three-dimensional scatterplots.

To reduce dimension we project the data from the many-dimensional data space to a low-dimensional view. In general, the projection could be any mapping from the many-dimensional data space to a low-dimensional view. The projection pursuit methods that have been developed so far restrict the mappings considered to orthogonal projections of the data onto one-, two-, or three-dimensional subspaces of the euclidean data space.

Until recently, work on projection pursuit has concentrated on automatic algorithms. In automatic projection pursuit, a numerical measure of interesting structure is chosen. Then an optimization algorithm chooses a projection to maximize this numerical criterion.

The first projection pursuit algorithm [30] used a numerical optimizer to search for a one- or two-dimensional projection that maximized a "clottedness" index, which was intended as a numerical measure of interesting structure.

Automatic projection pursuit methods have been developed for more well defined problems: regression [27,28], classification [28], and density estimation [28,25,37]. In these problems, we construct a model that summarizes the apparent dependence in our data set of a response on some predictors. In regression the response is one of the euclidean variables; in classification the response is a categorical variable; in density estimation, the response is the "mass" of the observations.

These projection pursuit models are aimed at a particular kind of structure: the nature of the dependence of the "response" on the predictors. The model is constructed not only to summarize this dependence, but also to discover it. In other words, projection pursuit models are used for exploration, as well as summary.

Projection pursuit models describe data following the idea expressed in Tukey's "Data = Fit + Residuals" [46,42]. The idea is to first explore to discover some structure, model the structure to summarize it, subtract (in some sense) the model from the data to get a set of residuals, and then repeat the process on the residuals to discover and summarize and remove further structure. In projection pursuit, this results in a model that is built up from several low-dimensional views of the data. The process is summarized by the <u>projection pursuit paradigm</u> [28]:

```
1.  Choose an initial model.

2.  Repeat

    a) Find a projection that shows deviation from the current
       model, indicating previously undetected structure
       (Projection Pursuit).

    b) Change the model to incorporate the structure found in
       a) (Model Update).

3.  Until the current model is a sufficiently accurate summary
    in all projections.
```

Projection pursuit methods have a natural connection to interactive graphics. As long as we project on a subspace of dimension no greater than three, we can look at a picture of the result. An interactive graphics system, like Orion I, lets a data analyst modify or take over some functions performed by machine in the automatic versions of projection pursuit.

For example, the Update Projection command discussed in chapter III allows a user to search for an interesting projection, manually immitating the Rosenbrock search strategy [44] used by the numerical optimizer in the automatic versions. However, a human being can search to improve a subjective impression, using perception, judgement,and a knowledge of the context of the probelm, instead of relying on a single, numerical measure of what constitutes an interesting view.

5.2 PROJECTION PURSUIT REGRESSION

In this section I discuss projection pursuit regression. I describe the original, "batch" version. In this version, all decisions are made automatically, by machine, on objective, numerical criteria. In the next section, I will describe how projection pursuit regression can benefit from human interaction.

5.2.1 Regression

In the context of regression, we distinguish one of the p_e euclidean variables as the response variable. The remaining $p_e - 1$ euclidean variables are called the predictor variables. We refer to the value of the response variable for the ith observation as y_i. The values of the predictor variables for the ith observation form a vector, $\vec{x}_i$.

One way of looking at the regression problem is to say that we are looking for a function or model, $f(\cdot)$, that summarizes the apparent dependence, in the data set at hand, of the responses, $\{y_i\}$ on the predictors, $\{\vec{x}_i\}$. We have:

$$y_i = f(\vec{x}_i) + r_i$$

$\{r_i\}$ are residuals from the model. There are two things that we want from a regression model:

1. The model, $f(\cdot)$, should be "nice" or "simple".

2. The residuals, $\{r_i\}$, should be "small".

These two goals are usually conflicting.

In parametric regression we assume that the form of the model, $f(\cdot)$, is known and $f(\cdot)$ is completely determined by a small number of parameters, which are to be estimated.[5] Our model is "nice" because we

[5] Unfortunately, the word "parametric" is used to refer to two dis-

constrain it to be a member of a "nice" parametric family of models. We make the residuals small by choosing values of the parameters to minimize some measure of the size of the residuals. One common criterion for small residuals is least squares, that is, $f(\cdot)$ is chosen to make $\sum_i r_i^2$ small.

Non-parametric regression makes no (parametric) assumptions about the functional form of the dependence. To make the regression model "nice", we require it to be "smooth". "Smooth" is taken to mean that observations that are close in the predictor space have similar values of $f(\cdot)$. The function, $f(\cdot)$, should vary slowly relative to the spacing of the observations in the predictor space.

Most methods for non-parametric regression are "local averages". That is:

$$f(\vec{x}_i) = \text{AVE} \{ y_j : \vec{x}_j \in N(\vec{x}_i) \}$$

These methods differ in the sense in which the local neighborhood, $N(\vec{x}_i)$, is defined and in the notion of average, AVE, that is applied to observations in the local neighborhood. Two examples of local average regression are k-nearest neighbor regression [45] and recursive partitioning regression[23].

The difficulty that many local average algoirthms run into is the "curse of dimensionality" [5]. This phrase refers to the fact that in many-dimensional spaces the data are inherently sparse. Either the local neighborhoods are very large or they contain very few points. One of the principal motivations for projection pursuit methods, regression included, is to avoid the curse of dimensionality.

5.2.2 Linear regression

Projection pursuit regression can be described from first principles. However, it seems to be most easily accepted by statisticians if it is introduced as a generalization of the familiar linear model. In linear models, $f(\cdot)$ is restricted to be a linear (actually affine) functional of the predictors. In non-standard notation:

---

tinct sorts of parametric assumptions about regression models. I use "parametric" to refer to assumptions about the form of the regression model -- linear, logistic, exponential, polynomial,sinusoidal, or whatever. "Parametric" also commonly refers to assumptions about a probability model for errors from a hypothetical true deterministic regression model.

$$f(\vec{x}_i) = a + \langle \vec{w}, \vec{x}_i \rangle = a + \sum_j w_j \cdot x_{ij}.$$

Traditionally, the parameters of the linear model, a and $\vec{w}$, are chosen by least squares, that is, to minimize

$$RSS = \sum_i (y_i - a - \langle \vec{w}, \vec{x}_i \rangle)^2$$

Note that $f(\cdot)$ depends on $\vec{x}_i$ only through its inner product with $\vec{w}$.

5.2.3 Another ~~v~~iew of linear regression

To connect linear regression to projection pursuit regression, it is convenient to re-write the linear model:

$$\begin{aligned} f(\vec{x}_i) &= a + \langle \vec{w}, \vec{x}_i \rangle \\ &= a + \|\vec{w}\| \cdot \langle \vec{w}/\|\vec{w}\| , \vec{x}_i \rangle \\ &= a + b \cdot \langle \vec{u}, \vec{x}_i \rangle, \end{aligned}$$

where a and b are real numbers and $\vec{u}$ is a unit vector, that is,

$$\|\vec{u}\| = 1.$$

As above, a, b, and $\vec{u}$ are determined by least squares, subject to $\|\vec{u}\|=1$.

Note that this model depends on the $\vec{x}_i$ only through the inner product $\langle \vec{u}, \vec{x}_i \rangle$. In other words, the model depends on the predictors only through their projection on the direction, $\vec{u}$.

This formulation suggests a way to look at a picture of a linear regression. Consider a two-dimensional scatterplot of $\{y_i, z_i\}$ where $z_i = \langle \vec{u}, \vec{x}_i \rangle$. This is a picture of the data as projected on the two-dimensional plane spanned by the response variable and the direction, $\vec{u}$, in the space of the predictor variables. The model is:

$$f(\vec{x}) = a + b \cdot \langle \vec{u}, \vec{x} \rangle = a + b \cdot z$$

So, in this projection, the values of the model lie on the line,

$$y = a + b \cdot z.$$

5.2.4 Smooth to generalize straight line

By comparing the cloud of points, (y_i, z_i), to the line, $a + b \cdot z$, we can judge, graphically, how appropriate the linear model is. If the line fails to capture the dependence of y_i on z_i, then we need to generalize the linear model.

For the moment, suppose we hold $\vec{u}$ fixed. Then the question is how to summarize the dependence of the responses, y_i, on a one-dimensional predictor, in this case, $z_i = \langle \vec{u}, \vec{x}_i \rangle$. This is just a one-dimensional version of the general non-parametric regression problem described above. We want to summarize the response by a non-linear function and we do not want to make parametric assumptions about the form of the non-linear function. With a one-dimensional predictor space there is no problem with the curse of dimensionality. So we are free to use some form of local averaging. In one-dimensional problems, methods based on local averages are refered to as smoothers. The non-parametric regression problem with a one-dimensional predictor space is sometimes refered to as smoothing scatterplots [29].

Existing versions of projection pursuit regression use a running linear fit as a smoother [11,29]. The running linear fit has a parameter called the span, which is the number of observations in the local neighborhood. The smoothed value is determined by fitting a linear model to the local neighborhood. The smoothing algorithm we use is described in detail by Friedman and Stuetzle in [27,29].

5.2.5 Choosing a direction

Given a smoothing algorithm, we choose the direction $\vec{u}$, so that the smooth along $\vec{u}$ minimizes some objective criterion, for example, least squares. That is, we choose $\vec{u}$ to minimize

$$RSS = \sum_i (y_i - g(\langle \vec{u}, \vec{x}_i \rangle))^2,$$

where $g(\cdot)$ is the function chosen by the smoothing algorithm.

To find the $\vec{u}$ that minimizes RSS, we use a modified Rosenbrock method for optimization [44]. The Rosenbrock method is an old (1960) and fairly naive method. The basic idea is to minimize a function of several variables by optimizing over one variables, while holding all others fixed. The method we actually use is slightly simpler in some respects than Rosenbrock's original proposal and is modified to restrict the search to the unit sphere. To minimize a function, $R(u_1, u_2, \ldots, u_p)$:

```
1.  Choose a starting value for $\vec{u}$

2.  Repeat

    a) For i = 1 to p do

       i)  Minimize $R(u_1,u_2,\ldots,u_p)$ by varying $u_i$, renor-
           malizing to keep $\vec{u}$ a unit vector.

3.  Until the reduction in $R(\vec{u})$ meets some convergence cri-
    terion.
```

More sophisticated optimization algorithms may increase the computational efficiency of automatic versions of projection pursuit. However, because the Rosenbrock method varies only one parameter at a time, it can be executed manually in a natural way, which may not be true for more sophisticated optimizers.

5.2.6 More complicated models

Once the optimizer has chosen a direction, $\vec{u}$, our model is:

$$f(\vec{x}_i) = g(\langle\vec{u},\vec{x}_i\rangle).$$

The model depends on the predictor variables only through their projection on $\vec{u}$. In a sense, the model varies only in the direction, $\vec{u}$, because the model is constant along all directions orthogonal to $\vec{u}$.

To model more general regression surfaces, we iterate the above procedure. That is, following the projection pursuit paradigm [28], we subtract the current model from the responses to get a set of residuals. We then search for a direction and smooth function to fit the residuals. To summarize:

```
1. Choose an initial model:
       $f^{(0)}(\vec{x}_i) = (1/n) \sum y_i.$

2. Repeat
   a) Form residuals:
          $r_i^{(k)} = y_i - f^{(k-1)}(\vec{x}_i).$
   b) Choose $\vec{u}$ to minimize:
          $RSS^{(k)} = \sum_i (r_i^{(k)} - g^{(k)}(\langle\vec{u}^{(k)},\vec{x}_i\rangle))^2$
   c) Update the model:
          $f^{(k)}(\vec{x}_i) = f^{(k-1)}(\vec{x}_i) + g^{(k)}(\langle\vec{u}^{(k)},\vec{x}_i\rangle)$
   d) Increment $k = k + 1$;

3. Until the percentage reduction in $RSS^{(k)}$ is below some
   threshold.
```

I am ignoring many details of the automatic procedure, such as "backfitting". For a complete description see Friedman and Stuetzle [27,28].

5.2.7 The final model

To summarize, projection pursuit regression models a response variable as the sum of general smooth functions of linear combinations of the predictor variables:

$$f(\vec{x}_i) = \sum_k g^{(k)} (\langle \vec{u}^{(k)} , \vec{x}_i \rangle).$$

5.3 INTERACTIVE PROJECTION PURSUIT REGRESSION

5.3.1 Why interactive?

The original projection pursuit regression algorithm was designed for computational efficiency, accurate estimation, and adaptability to a wide variety of circumstances. Since these desirable qualities are usually antagonistic, the design is a compromise.

The motivation for writing an interactive version of projection pursuit regression is to overcome limitations imposed by unavoidable design compromises in the non-interactive projection pursuit regression. In addition, we learn much more about data by watching and actively participating in the construction of the regression model.

5.3.2 What to modify manually?

The interactive projection pursuit regression program (IPPR) permits a human being to modify or completely take over functions handled by automatic procedures in the original projection pursuit regression. I describe two important functions in detail:

1. searching for the next direction in the predictor space, along which to update the model.

2. choosing the smooth function on that direction.

5.3.3 Searching for a direction

In IPPR, the projection and the viewing transformation are restricted so that screen-y (vertical) always corresponds to the response variable. The screen shows, at each instant, a scatterplot of the response variable, y_i, versus the data projected on the current direction in the space of predictor variables, $z_i = \langle \vec{u}, \vec{x}_i \rangle$ (see Plate 7). We may choose to display the curve of the automatic smoother, which smooths y_i as a function of z_i. Also at our option, a vertical bar, whose height indicates the value of an objective criterion may be drawn on the bottom of the screen. The objective criterion that we use is the percentage of variance in the response explained by the smooth.

When we move the trackerball, the point cloud rotates about the y-axis (the response variable). The display then shows a three-dimensional scatterplot. The three-dimensional space is that spanned by the response variable and a two-dimensional subspace of the predictor variables.

As we rotate, the direction, $\vec{u}$, corresponding to screen-x changes. The curve of the smoother is continuously updated, in real-time. This is an example of an intensive real-time computation that is made possible by our arithmetic processor, the 168/E, and was not possible on earlier Prim systems. Orion I can compute the rotation, erase and redraw the points, compute the new smooth, and erase and redraw the curve of the smoother at about 5 times a second for a data set with 500 observations.[6] This is fast enough for interactive searching.

The vertical bar indicating the value of the objective criterion is drawn at a horizontal position depending on the angle of rotation. As the data is rotated, the vertical bar traces out a bar-graph of the value of the objective criterion as a function of the angle of rotation. This makes it easy for us to manually rotate to the best direction in the current predictor plane. We can now update the projection to choose a new plane in the predictor variables.

The above suggests a manual search strategy that imitates the Rosenbrock method of numerical optimization used by the original projection pursuit regression:

```
1. Repeat

     a) Replace screen-z by one of the variables, so that the
        screen-x -- screen-z plane corresponds to a new plane
        in the predictor space, that spanned by $\vec{u}$ and the cho-
        sen variable.

     b) Rotate to find the "best" direction in the current
        plane.

2. Until there is no significant improvement in perceived
   goodness of fit.
```

[6] The time limiting part of the current system is the graphics device and not the speed of computation. Computing a new rotation and resmoothing takes about 1/4 of the time to draw a new picture.

Two aspects of the searching function can benefit from human judgment. First, we can sometimes shortcut the numerical optimizer. Prior knowledge of the data may suggest good starting values of the direction. Prior knowledge may also suggest which directions of search may be most useful. Second, the automatic search is limited by the need for a numerical objective criterion to optimize. We can search for a direction that is "good" in a subjective sense.

No single objective criterion is correct in all circumstances. The original projection pursuit regression uses the sum of squared residuals from the smooth as the objective criterion to optimize. This is not resistant to the effects of a few "bad" observations. There are many candidates for more resistant alternatives; most try to identify observations that have unusual or extreme values and give those observations less weight. It is usually not clear beforehand exactly what criterion is best. For example, a more resistant criterion may be desirable in choosing the first terms of the model and a less resistant criterion may be appropriate later.

Subjective aspects of the relationship between the current direction and the response variable can influence our choice of a direction for the next term in the model. For example, a direction involving a small number of variables may be preferred because it is easier to interpret, even though it has a slightly higher sum of squared residuals. In another case, a certain sub-optimal direction may be preferred because the shape of the smooth is simple. A combination of a "simple" direction with a "nicely" shaped smooth may be especially interesting because it suggests a parametric model for the dependence of the response variable on the predictors.

5.3.4 Choosing a smooth function

Given a direction in the predictor space, we have several ways of choosing the smooth function used in the next term in the model.

1. Accept the curve produced by the automatic smoothing procedure.
2. Adjust parameters in the automatic smoothing procedure, such as span, with commands that use the trackerball for input. By watching the curve change as, for example, the span is changed, we can see how the shape of the curve depends on the parameters in the automatic smoothing procedure.
3. Choose a particular smoothing algorithm from from a menu of alternatives.

4. Manually adjust the parameters in some parametric family of curves, such as Chebyshev polynomials.

5. Position, with the trackerball, several "knots" on the screen, which are joined by splines. The spline curve can be continuously updated, as knots are added, deleted, or moved. (This is another example of a parametric family of curves.)

6. Draw a curve freehand through the data, using the trackerball or, more naturally, a digitizing pen and tablet.

Only methods 1 and 2 are available at present.

5.3.5 Connection to Grouping

Another way in which IPPR benefits from interaction is by combining IPPR with the Group and Select Category commands discussed in chapter III. The Group command allows us to partition the data set into subsets. We may set some subsets inactive with the Select Category command and proceed to fit the regression model to the remaining, active observations. This provides a convenient way to compare models fit on all the data to models fit on various subsets.

5.3.6 Assessing the variability of the smoother

When we search manually for a direction, we often sacrifice some amount of the objective, numerical criterion of "goodness of fit" to improve some subjective impression. For example, we may prefer a direction that has non-zero coefficients of as few predictor variables as possible. To achieve a parsimonious model, we may be willing to sacrifice a few percent of variance explained.

To judge how important a few percent of variance is, we need some indication of the natural variability of our regression model. In a non-interactive setting, it would be appropriate to try to assess the variability of the full projection pursuit algorithm. This could be done by bootstrapping [17,18,19,20] the procedure (see Plate 8).

In an interactive program, we need a way of assessing variability that can be done rapidly, in real time, and that has a natural graphical representation. A reasonable solution is to bootstrap the smoother only. That is, we hold the direction, $\vec{u}$, fixed so that have a two-dimensional scatterplot of y_i vs. $z_i = \langle \vec{u}, \vec{x}_i \rangle$. We then bootstrap the smooth of y_i as a function of z_i.

We may be willing to give up a little "goodness of fit" in our interactive choice of the direction, $\vec{u}$, if the difference in "goodness of fit" between a parsimonious model and the "best" direction is small compared to a natural measure of the variability in "goodness of fit" associated with the smoothing process.

There are two alternatives for bootstrapping a smoothing algorithm: resampling observations and resampling residuals.

We resample observations by choosing observations uniformly at random from our data set, with replacement, to construct a bootstrap sample, $\{(y_i, z_i)^*\}$. The bootstap sample may also be viewed as a set of random weights for each observation in the data set. The smoothing algorithm is then applied to the bootstrap sample. Choosing repeated, independent bootstrap samples and looking at the variation in the smooths over the collection of bootstrap samples provides a natural indication of the variability of the smoothing algorithm.

To resample residuals, we subtract the smooth, $g(z_i)$, from the responses, y_i, to get a set of residuals, $\{ r_i = y_i - g(z_i) \}$. Then, for each observation, we choose at random, uniformly, with replacement, one of the residuals, r_i^*, to add to the smoothed response, $g(z_i)$, for that observation to get a pseudo-response value:

$$y_i^* \quad = \quad g(z_i) \; + \; r_i^*.$$

We then smooth the resulting set $\{(y_i^*, z_i)\}$. We repeatedly construct sets of pseudo-responses by sampling from the original residuals, r_i, and adding them to the original smooths, $g(z_i)$, to create a collection of bootstrapped smooths, $g^*(z_i)$.

If there is greater variation about the smooth in one region than in others (heteroscedasticity), resampling residuals uniformly will smear out the variation over all the observations. For some purposes it may be important to know accurately the local variability of the smooth. To allow this, we provide an option for local resampling of residuals. In local resampling, a residual is chosen from those corresponding to observations that fall in a window about the observation whose smoothed response the resampled residual is to be added to. We take the window to be the same as the span of the smoothing algorithm.

IPPR provides two options for viewing the results of bootstrapping: fill and wiggle. When the bootstrap command is chosen, the program resamples, computes, and draws new smooths in real time. This process is executed about 5 times a second for a data set with 500 observations. Thus 100 bootstrap replications will be seen in about 20 seconds.

In the fill option, the curves of all replications of the smooth accumulate on the screen. The original smooth is redrawn each time in a distinct color, so that it can be compared with the bootstrap replications. The smooths of the bootstrap samples soon fill in a "confidence band" about the original curve, and give an indication of the variability of the smoothed value for any value of z.

In the wiggle option, only the last replication of the curve is drawn. This makes it possible to assess the variability of features of the curve, such as sharp bends, which are obscured in the band of color produced by the fill option.

Both options also accumulate on the screen a histogram of the values of the objective criterion for the collection of bootstrap replications. The value of the objective criterion for the original curve is marked on the histogram, so that it may be compared to the variation under resampling. If the spread in the histogram is large, we are unlikely to worry much about sacrificing a small amount of "goodness of fit" to achieve a parsimonious model.

Chapter 6

MULTIPLE VIEWS

In projection pursuit, we develop a model of many-dimensional structure from a_ sequence of low-dimensional views. To interpret a projection pursuit model, we need to understand the relationships between the contents of two or more views in the sequence. The methods described in this section allow us to do this.

This approach is inspired by the M-and-N-plots of Diaconis and Friedman [14].

6.1 M-AND-N-PLOTS

A two-and-two-plot is one kind of M-and-N-plot. Two-and-two-plots are used to display four-dimensional data. To make a two-and-two-plot, we draw two two-dimensional scatterplots side by side. The two scatterplots show different pairs of variables. For each observation, there is one point in each of the two scatterplots. We then connect corresponding points in the two scatterplots by lines. To get a picture that is not confusing, only a subset of all possible lines is actually drawn. Diaconis and Friedman give an algorithm based on minimal spanning trees for deciding which lines to draw.

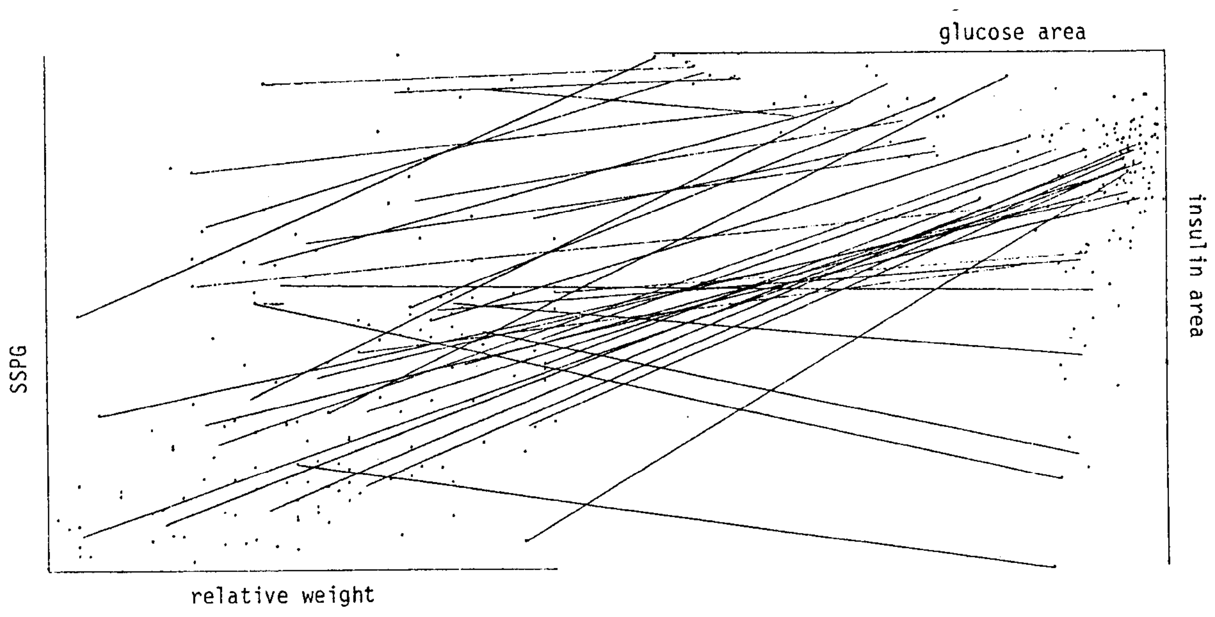


Figure 3: An M and N plot from [14].

6.2 <u>COLOR</u> <u>M-AND-N-PLOTS</u>

My idea is a modification of the above. Instead of connecting corresponding points by lines, I draw corresponding points in the same color (see Plate 9). The coloring is determined interactively.

A simple version of the program works as follows:

On the screen there are two scatterplots, side by side,showing four variables. There is a cursor on the screen in one of the two scatterplots. The scatterplot that the cursor is in is the <u>active</u> <u>scatterplot</u>. We position the cursor in the active scatterplo by moving the trackerball. Points near the cursor in the active scatterplot are red. Points at an intermediate distance from the cursor are purple. Points far from the cursor are blue. A point in the non-active scatterplot are given the same color as the corresponding point in the active scatterplot. The colors are continuously updated as the cursor is moved. We can also move the cursor from one scatterplot to the other, changing which scatterplot is active.

Using color instead of line segments to connect points has a disadvantage; it is not as precise at showing us the connection between a pair of points representing the same observation. However, we are not usually very interested in single observations. More often, we want to see how a region in one scatterplot maps into the other scatterplot; the combination of local coloring and the moveable cursor is a good way of seeing regional relationships between the two scatterplots.

Using color instead of lines makes it is possible to look at more than two scatterplots at once. Connecting corresponding points with line segments in more than two scatterplots at a time would produce a hopelessly confusing picture. With color, on the other hand, it is no more difficult to look at three or more scatterplots at once than it is to look at two.

6.3 OTHER COLORING SCHEMES

It is useful to have other options for determining the colors in the active scatterplot. For example, instead of determining color by distance from the cursor, the colors can be determined by the vertical (or horizontal) position of the cursor. Points above the cursor will be in one continuous range of hues, from blue through magenta to red. At the position of the cursor there is a discrete jump in hue, so that points below the cursor have hues from cyan through green to yellow.

6.4 CONNECTING THREE-DIMENSIONAL SCATTERPLOTS

So far I have discussed connecting the contents of two or more two-dimensional scatterplots. To get the full value of our graphics system we would like to connect two or more three-dimensional scatterplots.

There are two problems that prevent this from being completely straightforward. First, it is difficult to position a cursor in three dimensions. Second, it is difficult to control rotations and the motion of a cursor simultaneously. To get around these problems, I will provide two options:

In the first option, the coloring pattern is fixed and the rotation is under user control. The view space of the active plot is divided into eight octants: front-upper-right, back-upper-right, front-lower-right, etc. Each octant in the active plot has a distinct color. The corresponding point in the other plot(s) take the same color. All plots can be rotated together or each plot can be rotated seperately. As the points in the active plot rotate they will move from one octant to another, so the colors will change.

The division into octants is, of course, only one example of many possible coloring schemes.

The other option makes the rotations automatic and lets the coloring be determined by the position of the cursor. Because it is difficult to perceive the three-dimensional position of a cursor, the coloring should be determined by the positions of the points in the plane of the screen. Either all plots will rotate together or, possibly, only the non-active plots will rotate (it may be desirable for the plot that contains the cursor to remain fixed).

6.5 A VARIANT FOR GEOGRAPHICAL DATA

Color raster graphics devices have seen considerable use in the representation of geographical data by colored maps. We can combine maps with two- or three-dimensional scatterplots by extending the idea of Color M-and-N plots. A map is like a two-dimensional scatterplot; a region in the map, like a point in a scatterplot, represents an observation. Suppose we have a data set in which observations correspond to distinct geographical regions (for example, the Harrison-Rubinfeld housing data used in our two films [35, 6]). Then if we draw a map and a scatterplot side by side, we can connect points in the scatterplot with the corresponding regions in the map, using color and a cursor, just as we connected corresponding points in the Color M-and-N-plots.

For this to be effective, the colors of the points in the scatterplot and the regions in the map must be changed in real time, as the cursor is moved. Most reasonably priced raster graphics devices cannot, at present, redraw solid areas fast enough to do this in real time. However, we can change the colors of solid regions in real time if each region corresponds to a distinct address in the color look up tables (see chapter III). In the present Orion I system, this is possible only if there are no more than 255 regions in the map. However, a simple extension of the hardware, adding more bits per pixel to the frame buffer, would make it possible to change the color of 2^n regions in real-time, where n is the number of bits per pixel in the frame buffer.

Chapter 7

FUTURE RESEARCH WITH ORION I

I include this chapter to offer more evidence of the breadth, variety, and potential of new graphical methods in statistics. Also, the choice of directions for future research represents much of what we have learned in developing and using the programs discussed above. Finally, the ideas themselves are results of research; though inventing new methods is much less time consuming than writing programs to test them, much of the creativity in graphics research goes into inventing and choosing among new methods to work on.

7.1 IMPROVED SCATTERPLOTS

A user of Orion I looks at data by looking at scatterplots. The scatterplot, at least in two dimensions, is an old and successful tool in data analysis. However, it has some shortcomings when it is used to perceive the shape of the density in a point cloud. Scatterplots tend to reveal only crude differences in density, distinguishing regions of zero density from regions of non-zero density. Rather than seeing the shape of the density, we see the shape of something approximating the convex hull of the point cloud. This corresponds crudely to the support of the density, rather than the density itself. Perception of shape is therefore overly influenced by extreme points. Because it is not easy to see the difference between regions of high density and regions of low or moderate density, high density regions do not always receive the weight they deserve.

We can attack this problem in several ways with a color raster graphics device. We can color each point according to some estimate of the local density, so that points in high density regions will be bright and points in low density regions will be dim. This exaggeration may let us easily see finer distinctions than zero density versus non-zero density.

A less obvious alternative is to shade the background (the space between the points) according to local density, rather than the points themselves. This may let us see where our density estimate does not agree with the data.

7.2 EDGES

Another way of enhancing scatterplots is through the addition of edges. In other words, we draw line segments connecting some pairs of points in the scatterplot.

There are a few things that we hope to gain from including edges in the scatterplot. First, edges may aid our perception of the three-dimensional shape of the point cloud[16]. Second, a picture including edges may emphasize aspects of the shape of the point cloud that are not as striking in a pure point plot.

There are many ways to choose which edges to draw. In Prim-H [16], edges are chosen interactively; the user selects pairs of points with a cursor.

There are a number of sets of edges that are associated with a set of points in a natural way. One example is the minimal spanning tree [24]. We will try two versions of the minimal spanning tree: one based on distances in the current three space and one based on distances in the many-dimensional data space. The first alternative should be more useful for revealing the three-dimensional structure of the point cloud in the current view. The second alternative may be useful in searching for an interesting projection in an interactive projection pursuit. The minimal spanning tree based on distance in the data space will usually be very tangled when projected on an arbitrary three-dimensional subspace. However, if we find a view that untangles the tree, it is reasonable to suppose that this view captures at least some of the shape of the point cloud in the many-dimensional data space.

7.3 OTHER TYPES OF DATA

In classical multivariate analysis and in previous Prim systems it is assumed that each variable in the data sets to be examined is well represented by the real numbers. This is, of course, not always true. I have extended this definition to include explicit consideration of categorical variables. However, our methods for categorical variables are at present quite limited.

We have a good solution to the problem of looking at one categorical variable and three euclidean variables; we represent the three euclidean variables by the position of points in a three-dimensional scatterplot and the categorical variable by the color of the points.

For categorical variables with not too many categories, this works well.

We need to be able to look at several categorical variables at once. We do not know good methods that are effective for general combinations of categorical and euclidean varaibles. It will be necessary to understand better what "interesting structure" means for categorical variables. People have intuitive notions of what "interesting structure" is for euclidean variables, at least in three dimensions. But it is not obvious what the analogous ideas are for spaces of categorical variables or of mixtures of categorical and euclidean variables.

Work on special cases may give some insight to the general problem. For example, it is not to hard to look at the relationships between a two by two table (two categorical variables with two categories each) and three euclidean variables. We accomplish this by drawing four three-dimensional scatterplots, one for each cell of the two by two table. The four point clouds are rotated simultaneously, so that we can compare the shapes of the point clouds as they depend on the variables in the two by two table.

7.4 MISSING VALUES

The model of data described in chapter IV assumes that observations correspond to single points in the data space. This is often false in real data. Real data sets contain observations that have missing or censored values. One way of looking at this is to say that each observation defines not a single point, but rather a set in the data space.

The simplest way of dealing with missing variables is to delete all incomplete observations. One can also disregard variables that are unknown for a large number observations. However, especially in many dimensional problems, this can result in losing nearly all the data.

In the present system, we avoid throwing away data. We delete a point only while the current picture involves a missing variable. We never delete variables.

This is not completely satisfactory. In programs where we look at arbitrary linear combinations of variables, such as interactive PPR, we can soon find pictures where almost no points are left.

Another approach is to fill in the missing values. This is usually called imputation. We do not want to use parametric methods for imputation, because we do not know enough to build believable parametric models. A non-parametric method for imputing missing variables, based on recursive partitioning regression, has been developed by J.H. Friedman and W. Stuetzle. If we fill in all of the missing variables, we can proceed to analyze our data as though it had been complete to begin with.

We are likely to be uncomfortable with simply accepting the results of an imputation procedure, whether it is parametric or non-parametric. With Orion_I, we can study the performance of an imputation procedure and decide if its behavior seems reasonable.

A simple way to see what the imputation procedure does is to color points representing imputed observations differently from those representing observations that are completely known. We could then explore the data and see if the imputed observations seem to follow the pattern of the known data.

A good procedure for imputing missing variables will give us not only a guess for the missing values, but will also provide an estimate of the uncertainty of that guess. This would be useful information to have in the picture if we are trying to evaluate an imputation procedure.

For example, if an observation were missing a single variable, we would like our imputation procedure to give us an interval estima+e for the value of the missing variable. The incomplete observation would be represented in a scatterplot by a line segment that would show a set of reasonable guesses for the "true" position of that observation.

7.5 SHADED SURFACES

An advantage that raster graphics devices have over vector refresh displays is the ability to draw realistic looking solid objects by shading surfaces [22,43]. We will use shading to display regression surfaces to try to learn something about projection pursuit.

There are a number of open questions about projection pursuit. What sorts of functions can be approximated well by a few terms of a projection pursuit model? What sort of functions cannot be easily approximated? Given the set of directions used by a projection pursuit model, can we make any statements about where, in the predictor space, we expect the approximation to be good? Where should the approximation be poor?

These questions are difficult to address analytically. We may gain some insight by looking at pictures of a variety of concrete examples.

If we restrict ourselves to problems with a two-dimensional predictor space, then we can draw a picture of a regression surface by shading. The plane of the screen will corresponds to the two-dimensional predictor_ space. The height of the regression surface perceived through shading.

To gain insight into how a complex regression model is built we will look at a sequence of pictures. In the first picture we will show the result of smoothing along one direction. The second picture will show the smooth of the residuals along the second direction. the third picture will show the surface resulting from the sum of the first two terms. And so on.

7.6 PERIODIC BEHAVIOR IN TIME SERIES.

In this final section, I describe a method for analyzing periodic behavior in time series.

Suppose we have a time series X_t. We want to summarize X_t with a sum of a few (say 3 or 4) periodic functions:

$$X_t = \sum_i \phi_i(t) + r_t$$

The functions ϕ_i are each periodic with some period λ_i. As in the general regression problem, we want the ϕ_i to be "nice", we want the r_t to be small, and we also want there to be as few terms in the sum as possible.

A traditional approach to this problem is to compute the Fourier transform of the X_t's and look for peaks in the periodogram [8]. Each peak in the periodogram suggests including in the sum a sine with the appropriate phase and period. In this approach, the ϕ_i are nice functions because they are sinusoidal.

Fourier analysis has disadvantages. The presence of a few outliers can smear out peaks and make them invisible. In other words, "statistical" noise is not always high frequency. A periodic function which is not sinusoidal in shape may contribute many peaks to the periodgram and thus many terms to the sum. Even worse, if the series is the sum of only two periodic, but not sinusoidal functions, it may be impossible to seperate the effects of the two functions in the periodgram and the essentially simple nature of the series may be missed.

I propose a simple graphical method that should be a valuable complement to traditional Fourier analysis.

Let λ be a trial value for the period. We draw on the screen a (two-dimensional) scatterplot of X_t versus (t mod λ). The value of λ is changed by turning a dial (for us the trackerball). The scatterplot will change smoothly as we change the value of λ. If the series is indeed well summarized by the sum of a few periodic functions, then at a proper value of λ, the scatterplot will be closer to smooth curve.

To aid our perception, we will copy the regression program and draw a curve over the scatterplot by smoothing X_t as a function of (t mod λ). The curve will also change smoothly as we change λ. If we include a bar the measures some criterion of fit, such as percentage of variance explained, and let the horizontal position of the bar be determined by λ, then, as we vary λ, we will trace out a bar graph that is a generalized version of the periodogram. By choosing values of λ corresponding to peaks in this generalized periodgram, we can build up a model of the series.

We will usually start at large values of λ and gradually move to smaller ones. It is possible that the first period that fits well is in fact an integer multiple of the period we want. We can reduce an integer multiple to the true period by providing another way of changing λ. What we do is decrease λ in discrete steps, λ, $\lambda/2$, $\lambda/3$, ...

Once we have chosen a value of λ, we will subtract the smooth from the X_t to get a set of residuals R_t^1. We will repeat the process on the residuals to build up the model.

The method I am proposing has a similarity to projection pursuit. In a vague sense, it is a projection pursuit. For a fixed value of λ, the smooth is approximately the projection, in an L_2 sense, of X_t on the space of all functions that are periodic with period λ.

REFERENCES

1. Andrews, D.F., "Plots of High Dimensional Data", Biometrics 28, 1972

2. Andrews, D.F., "Statistical Applications of Real-Time Interactive Graphics", in Interpreting Multivariate Data V. Barnett, ed., J. Wiley and Sons, 1981.

3. Barnett, V., ed., Interpreting Multivariate Data, J. Wiley and Sons, 1981.

4. Bechtolsheim, A. and Baskett, F., "High-Performance Raster Graphics for Microcomputer Systems", Proceedings 1980 SIGGRAPH Conference, published as Computer Graphics 14 (3), July 1980.

5. Bellman, R.E., Adaptive Control Processes, Princeton University Press, 1961.

6. Belsey, D.A., Kuh, E., and Welsch, R.E., Regression Diagnostics John Wiley and Sons, 1980.

7. Beniger, James R. and Robyn, Dorothy L., "Quantitative Graphics in Statistics: A Brief History", The American Statistician 32 1978.

8. Bloomfield, P., Fourier Analysis of Time Series: An Introduction, John Wiley and Sons, 1976.

9. Box, G.E.P. and Cox, D.R., "An Analysis of Transformations", JRSS B 26, 1964.

10. Chen, Z. and Li, G., "Robust Principal Components and Dispersion Matrices Via Projection Pursuit", Research Report PJH-8, Dept. of Statistics, Harvard University.

11. Cleveland, W.S., "Robust Locally Weighted Regression and Smoothing Scatterplots", JASA 74, 1979.

12. Cleveland, W.S. and Terpenning, I.J., "Graphical Methods for Seasonal Adjustment", JASA 77, 1982.

13. Cleveland, W.S., Diaconis, P., and McGill, R. "Variables on scatterplots look more highly correlated when the scales are increased", Tech. Report #182, Dept. of Statistics, Stanford University, January, 1982.

14. Diaconis, P., and Friedman, J.H. "M and N Plots", Tech. Report #151, Dept. of Statistics, Stanford University, April, 1980.

15. Diday, E., et al., eds., Data Analysis and Informatics, North Holland, 1980.

16. Donoho, D., Huber, P.J., and Thoma, H., "The Use of Kinematic Displays to Represent High Dimensional Data", Research Report #PJH-5, Dept. of Statistics, Harvard University, March 1981.

17. Efron, B., "Bootstrap methods: another look at the jackknife", Ann. Stat. 7, 1979.

18. Efron, B., "Computers and the theory of statistics: thinking the unthinkable", Siam Review 21, 1979.

19. Efron, B., "Censored data and the bootstrap", J. Amer. Statist. Assoc. 76 1981.

20. Efron, B., "Nonparametric estimates of standard error: the jackknife, the bootstrap, and other resampling methods", Biometrika 1981.

21. Fisherkeller, M.A., Friedman, J.H., and Tukey, J.W. "Prim-9, An Interactive Multidimensional Data Display and Analysis System", Proc. 4th International Congress for Stereology, 1975.

22. Foley, J.D. and Van Dam, A., Fundamentals of Interactive Computer Graphics Addison-Wesley, 1982.

23. Friedman, J.H., "A tree structured approach to non-parametric multiple regression", in Smoothing Techniques for Curve Estimation, Gasser, T. and Rosenblatt, M., eds., Springer-Verlag, 1979.

24. Friedman, J.H. and Rafsky, L.C., "Multivariate Generalizations of the Wald-Wolfowitz and Smirnov Two-sample Tests", Ann. Stat. 7, p. 697-717, 1979

25. Friedman, J.H., Stuetzle, W., and Schroeder, Anne, "Projection Pursuit Density Estimation" Tech. Rep. Orion 002, Dept. of Statistics, Stanford University, July, 1981.

26. Friedman, J.H. and Stuetzle, W. "Hardware for Kinematic Statistical Graphics", Tech. Rep. Orion 005, Dept. of Statistics, Stanford University, November, 1981.

27. Friedman, J.H. and Stuetzle, W., "Projection Pursuit Regression", JASA v. 76, 1981.

28. Friedman, J.H. and Stuetzle, W., "Projection Pursuit Methods for Data Analysis", in Modern Data Analysis, Launer, Robert L. and Siegel, Andrew F., eds., Academic Press, 1982.

29. Friedman, J.H. and Stuetzle, W. "Smoothing of scatterplots", unpublished manuscript, May 1982.

30. Friedman, J.H. and Tukey, J.W. "A projection pursuit algorithm for exploratory data analysis", IEEE Trans. Comput. C-23 pp. 881-890, 1974.

31. Friedman, J.H., Tukey, J.W., and Tukey, P.A., "Approaches to analysis of data that concentrate near intermediate-dimensional manifolds", in Data Analysis and Informatics, Diday, E. et al., eds., North Holland, 1980.

32. Gnanadesikan, R. Methods for Statistical Data Analysis of Multivariate Observations, 1977.

33. Green, P.J. "Peeling Bivariate Data", in Interpreting Multivariate Data, V. Barnett, ed., J.Wiley and Sons, 1981.

34. Halmos, P.R., Finite Dimensional Vector Spaces, 2nd ed., Van Nostrand, 1958.

35. Harrison, D. and Rubinfeld, D.L. "Hedonic Prices and the Demand for Clean Air", Journal of Environmental Economics and Management, 5, 1978.

36. Huber, P.J., "Projection Pursuit", Research Report PJH-6, Dept. of Statistics, Harvard University.

37. Huber, P.J., "Density estimation and projection pursuit methods", Research Report PJH-7, Dept. of Statistics, Harvard University.

38. Kruskal, J.B., "Toward a practical method which helps uncover the structure of a set of multivariate observations by finding a linear transformation which optimizes a new 'index of condensation'", in Statistical Computation, R.C. Milton and J.A. Nelder, eds., Academic Press, 1969.

39. Kunz, Paul F., "The LASS hardware processor", Nuc. Instr. Meth. 9, p. 435, 1976.

40. Kunz, Paul F. et al., "The LASS hardware processor", in "Proc. 11th Annual Microprogramming workshop", SIGMICRO Newsletter 9, p. 25, 1978.

41. Launer, Robert L. and Siegel, Andrew F., eds., Modern Data Analysis, Academic Press, 1982.

42. Mosteller, F. and Tukey, J.W. Data Analysis and Regression, Addison-Wesley, 1977.

43. Newman, W.M. and Sproull, R.F., Principles of Interactive Computer Graphics 2nd ed., McGraw-Hill, 1979.

44. Rosenbrock H.H., "An automatic method for finding the greatest or least value of a function", Comput. J. 3, 1960.

45. Stone, Charles J., "Consistant nonparametric regression", Ann. Stat. 5, 1977.

46. Tukey, J.W. Exploratory Data Analysis, Addison-Wesley, 1977.

47. Tukey, J.W. and Tukey, P. "Graphical Display of Data Sets in 3 or more Dimensions", Part III, chapters 10-12 of Interpreting Multivariate Data V. Barnett, ed., J. Wiley and Sons, 1981.

48. Tukey, J.W. "Control and stash philosophy for two-handed, flexible, and immediate control of a graphic display", Technical Report No.221, Series 2, Department of Statistics, Princeton University, March 1982.