Statistical Science
1987, Vol. 2, No. 4, 355–395

Dynamic Graphics for Data Analysis

Richard A. Becker, William S. Cleveland and Allan R. Wilks

Abstract. Dynamic graphical methods have two important properties: direct manipulation of graphical elements on a computer graphics screen and virtually instantaneous change of the elements. The data analyst takes an action through manual manipulation of an input device and something happens in real time on the screen. These computing capabilities provide a new medium for the invention of graphical methods for data analysis. A collection of such methods—identification, deletion, linking, brushing, scaling, rotation, and dynamic parameter control—is reviewed. Those who develop dynamic methods must deal with a number of computer hardware and software issues, because for a dynamic method to work there must be a sufficiently fast flow of information along the channel that starts with the analyst's input and ends with the changed graph. Furthermore, for a dynamic method to be useful, the visual and manual tasks must be easy to perform. Several computing issues dealing with speed and ease of use—system bandwidth, input and output devices, low-level algorithms, device independent graphics, and data analysis environments—are reviewed.

Key words and phrases: Statistical graphics, computer graphics, brushing, rotation, multivariate analysis, software, hardware.

1. INTRODUCTION

Dynamic graphical methods have two important properties: direct manipulation and instantaneous change. The data analyst takes an action through manual manipulation of an input device and something happens, virtually instantaneously, on a computer graphics screen. Figure 1 shows an example in which a dynamic method is used to turn point labels on and off. The data analyst moves a rectangle over the scatterplot by moving a mouse; the figure shows the rectangle in a sequence of positions. When the rectangle covers a point, its label appears and when the rectangle no longer covers the point, its label disappears.

1.1 The Importance of Dynamic Methods

In the future, dynamic graphical methods will be ubiquitous. There are two reasons. One is the addition of dynamic capabilities to the methodology of traditional static data display provides an enormous increase in the power of graphical methods to convey information about data—wholly new methods become possible and many capabilities that are cumbersome and time consuming in a static environment become simple and fast (Tukey and Tukey, 1985). Huber (1983) aptly describes the importance of the dynamic environment: "We see more when we interact with the picture—especially if it reacts instantaneously—than when we merely watch." This does not mean that current static methods will be discarded, but rather that there will be a much richer collection of methods. The second reason is that the price and availability of powerful statistical computing environments are rapidly evolving in a direction that will permit the use of dynamic graphics (McDonald and Pedersen, 1985a, b). Thus, it seems likely that the methods described in this paper will be standard methodology in the future. Furthermore, because the number of people that have so far been involved in research in dynamic methods is relatively small, the development of new dynamic methods should accelerate as the appropriate computing environments become more widely available.

Richard A. Becker is a Member of the Technical Staff of the Statistics and Data Analysis Department, William S. Cleveland is Department Head of the Statistical Models and Methods Research Department, and Allan R. Wilks is a Member of the Technical Staff of the Statistical Models and Methods Research Department at AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, New Jersey 07974.

1.2 Two Early Systems

A recognition of the potential of direct manipulation, real-time graphics for data analysis goes back as far as the early 1960s when computer graphics systems

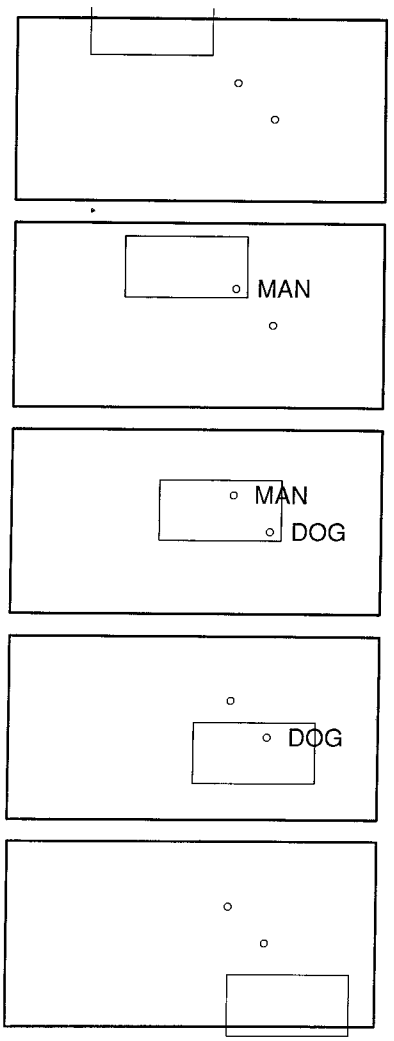

FIG. 1. *Dynamic graphics. In a dynamic graphic method, the data analyst directly manipulates graphic elements on the screen, causing virtually instantaneous changes of the elements. The figure shows an example. The analyst moves a rectangle around the screen by moving a mouse; when points enter the rectangle their labels appear and when the points leave the rectangle their labels disappear. The method used is brushing, which is discussed later.*

first began appearing. Tukey and Wilk (1965) write: "Visual presentation of $z = f(x, y)$ is far from easy, yet badly needed. Of three possibilities—contours, families of cross-sections, and isometric views—the first seems, so far, most likely to be effective, though direct-interaction graphical consoles may offer other possibilities"

In the late 1960s, Fowlkes (1971) developed a dynamic system for making probability plots. By turning a knob, the user could continuously change a shape parameter for the reference distribution on a probability plot displayed on the screen; the knob could be turned to make the pattern of points as nearly straight as possible. The data could be transformed by a power transformation, $(y + c)^p$, with c and p each changed by a knob. Points could be deleted by positioning a cursor on them. The system demonstrated that dynamic graphical methods had the potential to be important tools for data analysis.

Another early system was PRIM-9 (Fisherkeller, Friedman and Tukey, 1975), a set of dynamic tools for projecting, rotating, isolating, and masking multidimensional data in up to nine dimensions. Rotation was the central operation; this dynamic method allows the data analyst to study three-dimensional data by showing the points rotating on the computer screen. Isolation and masking were features that allowed point deletion in a lasting or in a transient way. PRIM-9 was an influential system; many subsequent systems were modeled after it and during the 1970s dynamic graphics and PRIM operations were nearly synonymous. (In fact, in the rush to implement PRIM systems, Fowlkes' ideas were nearly forgotten.) As the reader will see from the descriptions to follow and their origins, it was not until the early 1980s that significantly different methods would begin appearing; this was stimulated in large measure by new computing techniques coming from computer scientists.

1.3 Contents of the Paper

A variety of dynamic graphical methods are described and illustrated in Section 2 of this paper. Sections 2.1 to 2.6 cover identification, deletion, linking brushing, scaling, and rotation. Section 2.7 describes in a general way what many of the methods are doing—providing dynamic parameter control—and thereby opens the door to a large collection of potential methods. Computing issues are discussed in Section 3 of the paper; hardware and software considerations tend to be much more tightly bound to the success of dynamic methodologies than is the case for static graphical methods. Section 4 of the paper is a brief summary and discussion.

2. METHODS

2.1 Identification of Labeled Data

Identification has two directions. Suppose we have a collection of elements on a graph (e.g., points) and each element has a name or label. In one direction of identification we select a particular element and then find out what its label is; we will call this *labeling*. In the other direction, we select a label and then find the location on the graph of the element corresponding to this label. We will call this *locating*. Identification tasks, although seemingly mundane, are so all-pervasive that simple ways of performing them are of enormous help to a data analyst.

Labeling Points. Suppose x_i and y_i for $i = 1$ to n are measurements of two variables that have labels.

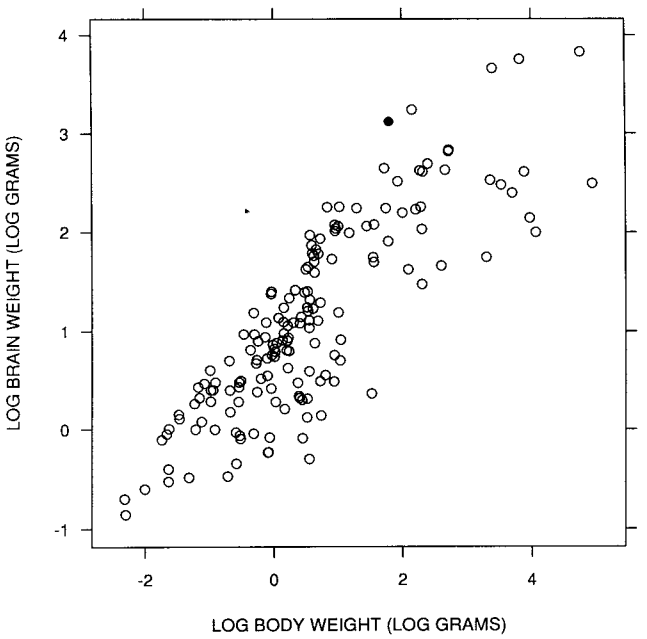

FIG. 2. *Identification. Log brain weight is graphed against log body weight for a collection of animal species. Each point has a name, the species name. As we saw in the previous figure, dynamic graphics provides easy methods for labeling: making the name for a point appear on the screen. We can also locate: find the data point with a particular name. In this figure, the species modern man was located by selecting it from a pop-up menu, which resulted in the circle for modern man being highlighted.*

Figure 2 shows an example. The data are measurements of the brain weights and body weights of a collection of animal species (Crile and Quiring, 1940). Biologists study the relationship between these two variables because the ratio (brain weight)/(body weight)$^{2/3}$ is a rough measure of intelligence (Gould, 1979; Jerison, 1955). In Figure 2, the data are graphed on a log scale and the axes are scaled so that $y = 2x/3$ is a 45° line; thus, 45° lines are contours of constant intelligence under this measure. Each point on the graph has a label: the name of the species.

In analyzing bivariate data with labels, we almost always want to know the labels for all or some of the points of the scatterplot. For example, in Figure 2 the point graphed by a *filled circle* is interesting because it lies on a 45° line that is above the lines for all other points, which means that this species has a higher intelligence measure than all others. Which is it? With static displays, finding out the labels of points on scatterplots is a cumbersome task. It is not possible to routinely show the labels of all points because overplotting frequently causes an uninterpretable mess; this would be the case in Figure 2. With dynamic displays, getting label information is simple because the data analyst can turn labels on and off very rapidly. One method for doing this has been illustrated in Figure 1. It turns out that the label for the interesting point in Figure 2 is, not surprisingly, "modern man."

Locating Points. In analyzing labeled points we often want to go in the other direction and locate a point with a certain name. For example, we might have asked where modern man is in Figure 2. This task also can be accomplished in a particularly efficient way by using dynamic graphics. The data analyst can press a button on a mouse to bring up a menu on the screen that shows the labels, move the mouse to select the label, and then release the button (Becker and Cleveland, 1987); this can result in the point being highlighted, as modern man is highlighted in Figure 2.

Locating Different Data Sets: Alternagraphics. In graphing two-variable data we often encounter another type of identification problem: the quantitative information is partitioned into subsets, and we want to locate the different subsets on the plot. For example, in Figure 2 the data can be categorized into five subsets: primates, nonprimate mammals, fish, birds, and dinosaurs. The subsets are shown in Figure 3. Another way to think of this is that there is a third variable, a categorical one, that we also want to show.

Subsets of the quantitative information on a graph can be enormously varied. Here are just three examples: 1) each subset is a set of points, as in Figure 3; 2) each subset is a collection of contours of a third variable as a function of the two axis variables; 3) the first subset is a scatterplot of points and the remaining subsets are regression curves of y on x resulting from several different models.

Comparing subsets can be surprisingly tricky when using static graphics. Part of the problem is that we want to be able to do more than just figure out to which subset each element of the graph belongs; we want to perceive each subset as a whole, mentally filtering out the others. With static graphics, many methods have been suggested (Cleveland, 1985)—use different plotting symbols, use different colors, or connect points of the same subset by lines. Often, none of these methods works, because the overlap of elements of the graph makes it impossible to distinguish the different subsets. In such cases the only solution is use juxtaposed panels as illustrated in Figure 3. The drawback to such a display is that we cannot compare the locations of different subsets as effectively as when all of the data are on the same panel.

A dynamic method that is often effective for identifying subsets is alternagraphics (Tukey, 1973). At a given moment in time the viewer can identify some of the subsets and the selection of identified subsets can be changed quickly. There are many ways of implementing this idea. One is to cycle through the subsets showing each for a short time period. More explicitly,

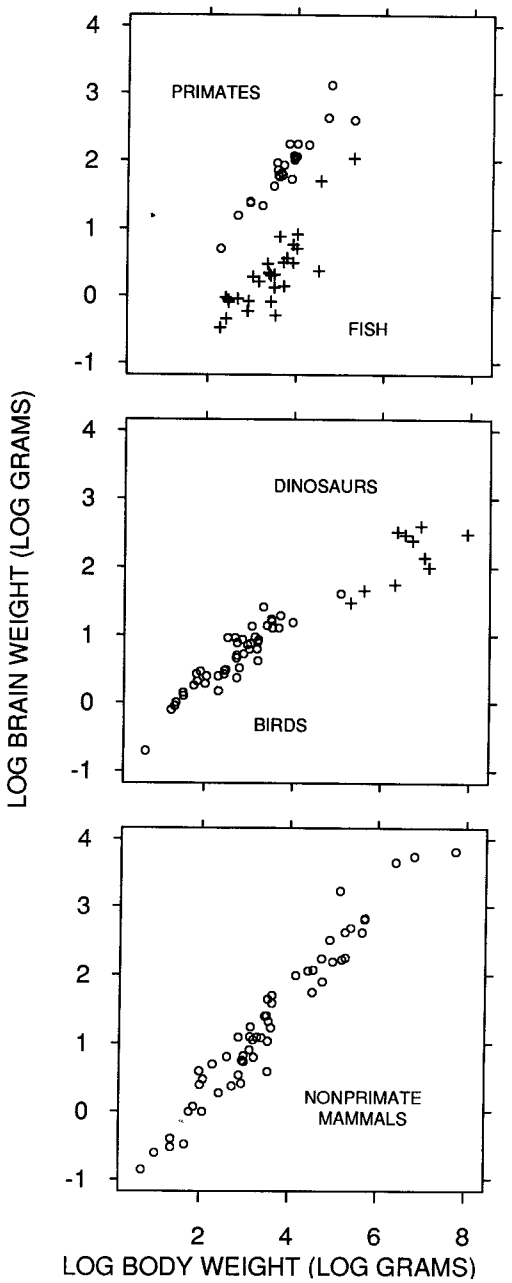

FIG. 3. *Alternagraphics. Another identification task for which dynamic methods are particularly well suited is locating different data subsets on a graph. For example, the brain and body data consist of five groups of species. In alternagraphics, data sets are identified by highlighting each in turn in rapid succession or by showing each set by itself and rapidly cycling through the sets.*

a graph of the first subset appears on the screen for, say, 1 sec, then it is replaced for 1 sec by a graph of the second subset, and so forth until we get to the last subset. Then the process repeats. Of course, the subsets are all shown on common axes so that the scale of the pictures remains identical as various subsets are shown. Another technique is to show all of the data at all times and have the cycling consist of a highlighting of one subset at each stage. A third technique is to provide the data analyst with the capability of turning any subset on or off with a simple and rapid action such as the use of a mouse to point to a subset name on a menu and then clicking a button; with such a technique the analyst can rapidly get any panel of Figure 3 (Donoho, Donoho and Gasko, 1985)

2.2 Deletion

Another fundamental operation that can be easily carried out by using dynamic graphics is deleting points from a graph. Figure 4 illustrates one simple use of deletion. A scatterplot is made and there is an outlier that causes the remaining points on the graph

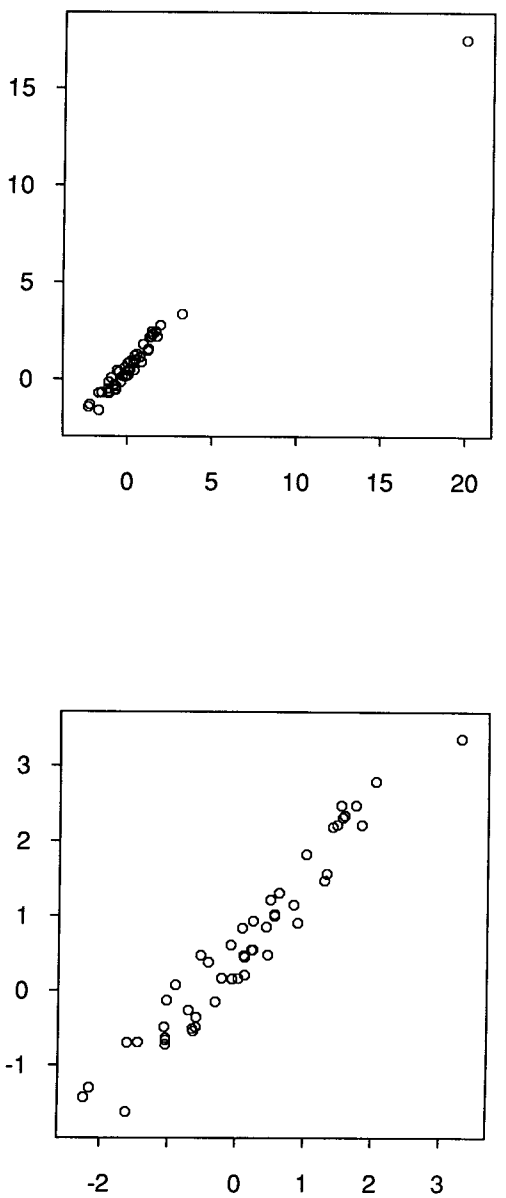

FIG. 4. *Deletion. In a dynamic graphics environment, points can be deleted by direct manipulation, for example, touching them with a cursor. In this figure, the outlier in the top panel is deleted, producing the graph of the bottom panel.*

to be crammed into such a small region that their resolution is ruined; the analyst removes the point by touching it with a cursor, and after the deletion the graph is automatically rescaled and redrawn on the screen. For example, Fowlkes (1971) used this dynamic deletion of outliers for probability plots; after points were selected for deletion, the expected order statistics of the reduced sample were recomputed automatically and the graph redrawn.

Deletion is actually a very general concept that can enter dynamic graphical methods in many ways. Its basic purpose is to eliminate certain graphical elements so that we can better study the remaining elements. For example, the outlier deletion lets us focus more incisively on the remaining data, and in alternagraphics, subsets can be temporarily deleted to allow better study of the remaining subsets. Other applications of deletion will be given in later sections.

2.3 Linking

Suppose we have n measurements on p variables and that scatterplots of certain pairs of the variables are made. A *linking* method enables us to visually link corresponding points on different scatterplots. For example, suppose there are four variables—x, y, w, and z—and that we graph y against x and z against w. To link points on the two scatterplots means to see by some visual method that the point (x_k, y_k) on the first plot corresponds to (w_k, z_k) on the other plot. To illustrate this, consider the Anderson (1935) iris data made famous by Fisher (1936). There are 150 measurements of four variables: sepal length, sepal width, petal length, and petal width. Two scatterplots are shown in Figure 5. The data have been jittered, that is, small amounts of noise added, to avoid the overlap of plotted symbols on the graph. Each scatterplot has two clusters, and we immediately find ourselves wanting to know if there is some correspondence between the clusters of separate plots.

Linking is a concept that has long existed in the development of static displays (Chambers, Cleveland, Kleiner and Tukey, 1983; Diaconis and Friedman, 1980; Tufte, 1983). One method for linking is the M and N plot of Diaconis and Friedman (1980); lines are drawn between corresponding points on the two scatterplots. Another method is to use a unique plotting symbol for each point (Chambers, Cleveland, Kleiner and Tukey, 1983) on a particular plot and to use the same symbol for corresponding observations on different plots.

A third method is the scatterplot matrix, all pairwise scatterplots arranged in a rectangular array, which arose, in part, because it provides a certain amount of linking. An example is Figure 6, which shows the iris data. To maximize the resolution of the plotted points, scale information is put inside the panels of the off-

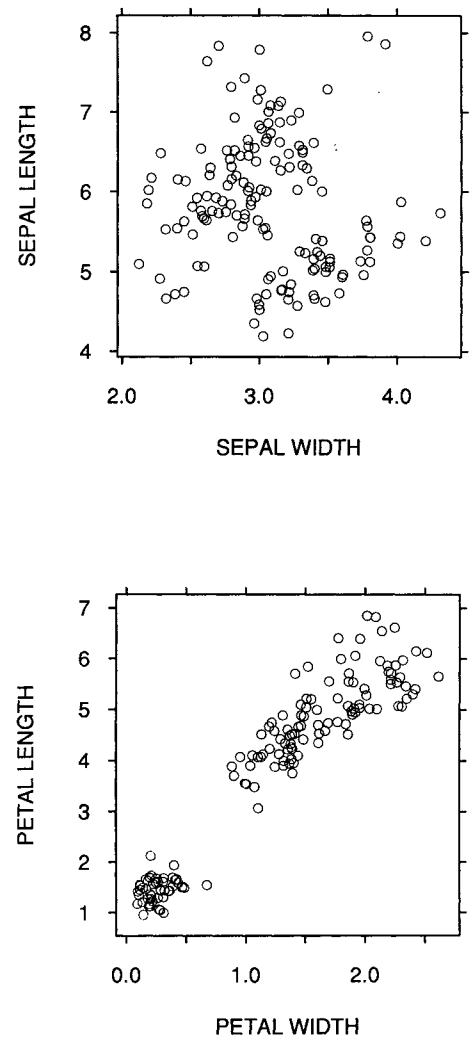

FIG. 5. *Linking. The scatterplots show the four variables of the Anderson iris data. Linking points on different scatterplots means visually connecting corresponding points.*

diagonal of the matrix; the labels are the variable names and the numbers show the ranges of the variables. Consider the cluster to the northwest in the sepal length and width plot of the (1,2) panel. Does this cluster correspond to one of the two clusters in the petal length and width scatterplot of panel (4,3)? By scanning horizontally from the (1,2) panel to the (1,3) panel and then vertically to the (4,3) panel we can see that the top half or so of the northwest sepal cluster corresponds to the top half or so of the northeast petal cluster. By scanning vertically from the (1,2) panel to the (4,2) panel and then horizontally to the (4,3) panel we can see that the left half or so of the northwest sepal cluster corresponds to the bottom half or so of the northeast petal cluster. The union of these two scans shows that most of the northwest sepal cluster corresponds to most of the northeast petal cluster; it is a good guess that the remaining pieces of the clusters also correspond.

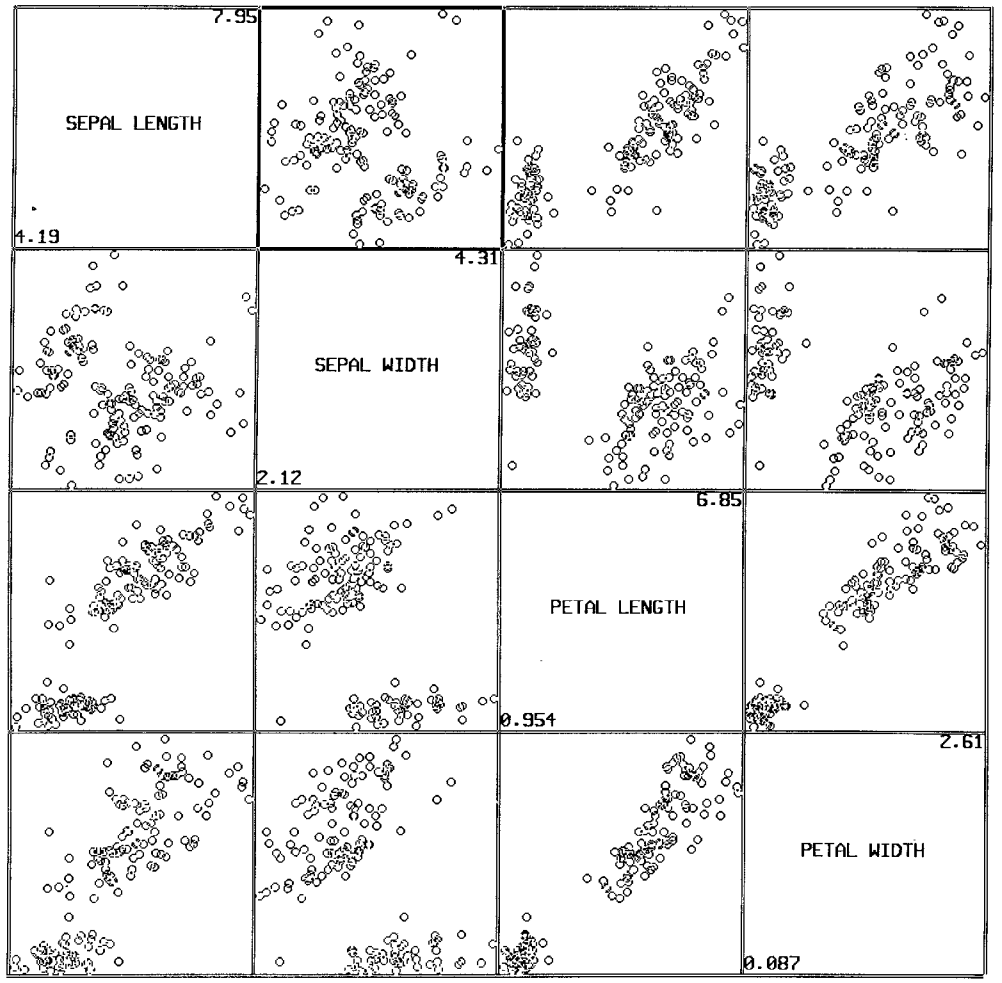

FIG. 6. *Linking. The iris data are shown by a scatterplot matrix, which allows a partial linking of points. Overlapping points on this screen dump sometimes give unusual effects, because of the XOR style of graphics being used (see Section 3.3).*

The earliest dynamic method for linking of which we are aware is described by Newton (1978). The analyst used a light pen to draw a polygon bounded by a set of points on one plot; the enclosed points, together with all corresponding points on other scatterplots, were highlighted. For example, in Figure 7, the points in the northwest sepal cluster of the (1,2) panel, as well as all corresponding points on other panels, have been highlighted. Now we can see easily that the northwest sepal cluster corresponds to the northeast petal cluster.

McDonald (1982) developed another dynamic system for linking. In his method a cursor is positioned on one plot. All points close to the cursor on the plot are red, all points that are intermediate distances from the point are purple, and all far points are blue. All points corresponding to the same observation on the collection of scatterplots have the same color. When the cursor is moved, the coloring changes to reflect the change in the cursor position.

2.4 Linking, Deleting and Labeling by Brushing

All of the methods for linking discussed in the previous section have certain limitations. The scatterplot matrix only allows partial linking. The M and N plot quickly becomes cluttered by the connecting lines. The unique plotting symbol method does not allow effective linking of groups of points and the overplotting of symbols obscures them. McDonald's method gives quite interesting results in some cases but has a limited domain of application because the coloring scheme does not provide a sufficiently incisive specification of regions of interest. Polygon drawing can provide incisive specification, but it is manually awkward and does not lead directly to a sufficiently rapid way of changing the region that is highlighted. In this section we will see how brushing overcomes these limitations.

Brushing is a dynamic method in which the data analyst moves a rectangle around the screen by mov-

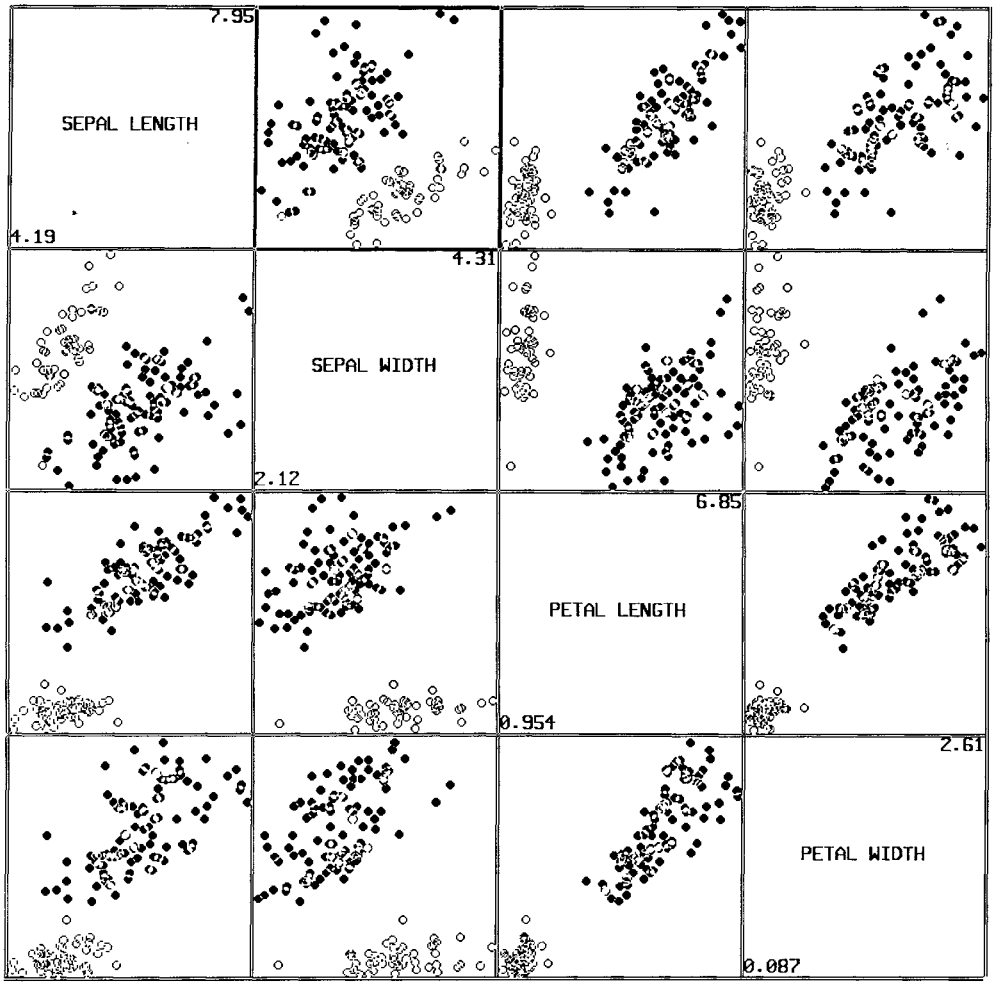

FIG. 7. *Linking. The northwest cluster in the* (1,2) *panel has been highlighted together with all corresponding points on the other panels. Now we can see that the northwest cluster in* (1,2) *corresponds to the northeast cluster of* (4,3).

ing a mouse (Becker and Cleveland, 1984 and 1987). The rectangle, which is called the *brush*, is shown in the (3,2) panel of Figures 8 and 9. Brushing allows linking in such a way that a wide variety of data analytic tasks can be carried out. Also, brushing can be used to delete and label points. In the next sections we will describe the features of brushing: movement of the brush, the ability to change the shape of the brush, three *paint* modes (transient, lasting, and undone), and four *operations* (highlight, shadow highlight, delete, and label).

The Scatterplot Matrix. Brushing can, in principle, be done on any subset of the collection of p $(p - 1)$ scatterplots of the p variables (Stuetzle, 1987). However, if we are interested in all p variables and in their relationships, it is usually best to operate on the entire scatterplot matrix. This matrix has been a successful static graphical tool because it provides a highly integrated view of all of the data. As we saw earlier, it is possible to carry out, purely visually, a partial linking of points on different scatterplots, even without brushing. Furthermore, by scanning a row or a column, we can see one variable graphed against all others in a convenient and efficient way. Of course, if p or n are too large it may not be possible to put the full matrix on the screen and have good resolution; in such a case we have to be satisfied with looking at subsets of the variables.

Highlighting, Brush Shape and the Three Paint Modes of Brushing. When the paint mode is *transient*, brushing with the *highlight* operation works as follows. A panel of the scatterplot matrix is selected as the active panel. All points on the active panel that are inside the brush are highlighted, as well as the points on other panels that correspond to these points. This is illustrated in Figure 8. The active panel is (3,2) and the brush is on this panel; it is covering the outlier on the sepal width and petal length scatterplot and so we can distinguish the points on other plots that correspond to the outlier. When the brush is moved, points

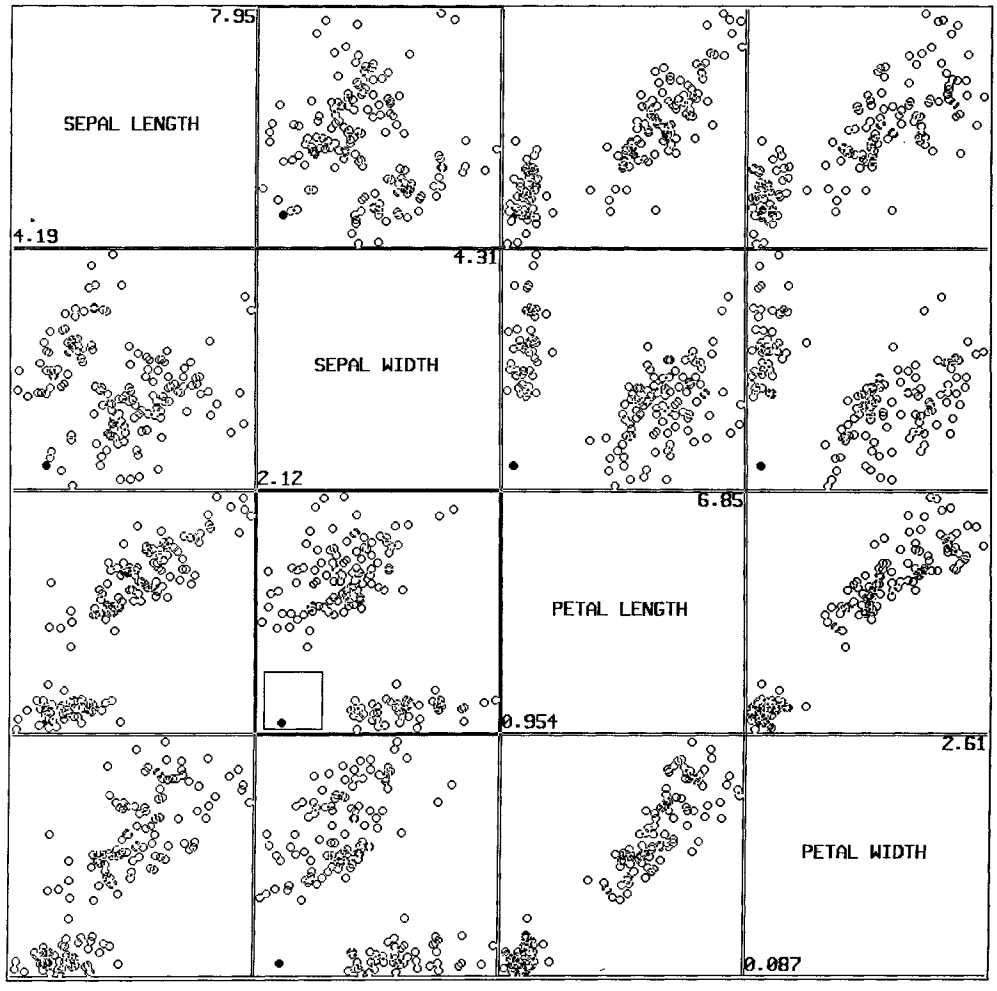

FIG. 8. *Brushing. Brushing is a dynamic method for labeling, deleting and linking by highlighting. The rectangle in the* (3,2) *panel is the brush, which the analyst moves around the screen by moving a mouse. When the operation is highlight, all points inside the rectangle on the active panel and all corresponding points on other panels are highlighted.*

that are no longer inside the brush are no longer highlighted and the new points inside the brush become highlighted; that is, the linked highlighting is transient. For example, in Figure 9 the brush has been moved to the right on the (3,2) panel and other points are highlighted.

Consider, on one of the panels of a scatterplot matrix, a subset of the points that cannot be enclosed in a rectangle without enclosing points not in the subset. With the transient highlighting just described, we cannot simultaneously highlight the points of the subset without also highlighting points not in the subset. An example is the northwest cluster in panel (1,2) of Figure 9. The problem is solved by the *lasting* paint mode: when one button of the mouse is pressed, points that are highlighted remain highlighted even after the brush no longer covers them. Thus, an irregular cluster can be highlighted by using a small brush and moving it over all sections of the cluster as if it were being painted. This is illustrated by the sequence in the first column of Figure 10. Lasting highlight can be removed with the *undo* paint mode; when another button of the mouse is depressed, the effect of the brush is to reverse the highlighting of the points inside it. This is illustrated by the sequence in the second column of Figure 10.

Usually, lasting highlight is most conveniently carried out when the brush is in the form of a small square or small rectangle; this helps prevent the slopping of highlight onto unwanted points (although undo makes it easy to fix mistakes). A small brush is also useful for highlighting a single point, as in Figure 8. Other brush shapes are useful in other situations, as we shall now see.

Conditioning on a Single Variable. By making the brush long and narrow, we can highlight points for which the values of a particular variable are in a narrow range; thus, we can study the relationships of

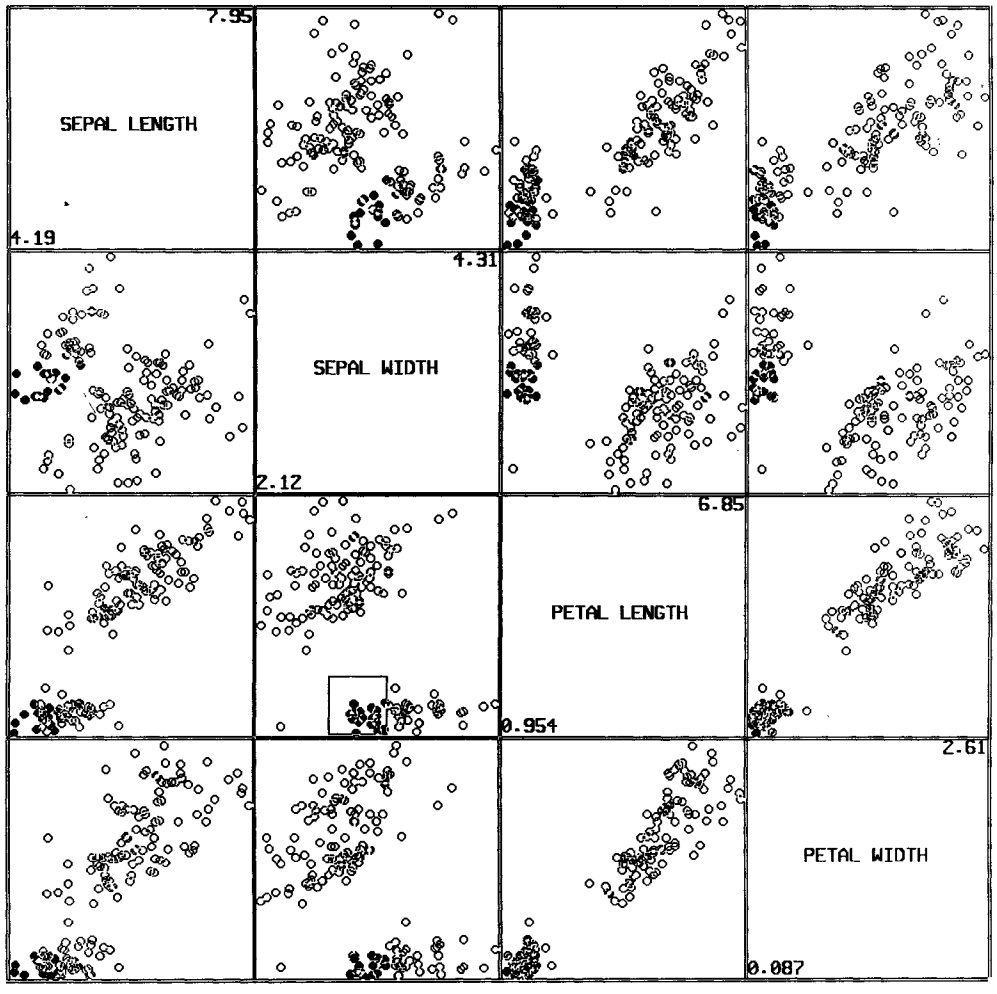

FIG. 9. *Transient paint mode. When the paint mode is transient, points previously inside the rectangle are no longer highlighted. In this figure the rectangle has been moved to the right from its position in the previous figure, and new points are highlighted.*

other variables for the selected variable held fixed to this range. This is illustrated in Figure 11. The data are from an experiment in which 30 rubber specimens were rubbed with an abrasive material (Davies, Box, Cousins, Hinsworth, Henny, Milbourn, Spendley and Stevens, 1957). The abrasion loss (the amount of rubber rubbed off), the hardness, and the tensile strength of each specimen were measured. The purpose of the experiment was to see how abrasion loss depends on hardness and tensile strength. In the original analysis of these data, a linear regression model was fit. In Figure 11 the brush is selecting low values of hardness; from the highlighting in the (3,2) panel we can see that there is a *nonlinear* dependence of abrasion loss on tensile strength for hardness held fixed to low values. Figure 12 shows a sequence consisting of just the (3,1) and (3,2) panels of the matrix. As we go from top to bottom in the figure, the brush is moving from left to right in the (3,1) panel, conditioning on higher and higher values of hardness. The highlighted points of the (3,2) panels generally form nonlinear patterns that appear to be just vertical translations of one another. Figure 13 is similar except that now we are conditioning on different ranges of tensile strength; the highlighted points of the panels on the left, which show the dependence of abrasion loss on hardness for tensile strength held fixed to the various ranges, also appear to be vertical translations of one another, but unlike those in Figure 12 they have a linear pattern.

Figures 12 and 13 suggest three things about the data. First, there is a nonlinear dependence of abrasion loss on tensile strength; the linear model used in the original analysis appears incorrect. Second, there is a linear dependence of abrasion loss on hardness. Third, because of the vertical translation of patterns in Figures 12 and 13, a model for the data does not need a hardness and tensile strength interaction term. These conclusions have been confirmed by regression modeling of these data (Cleveland and Devlin, 1986).

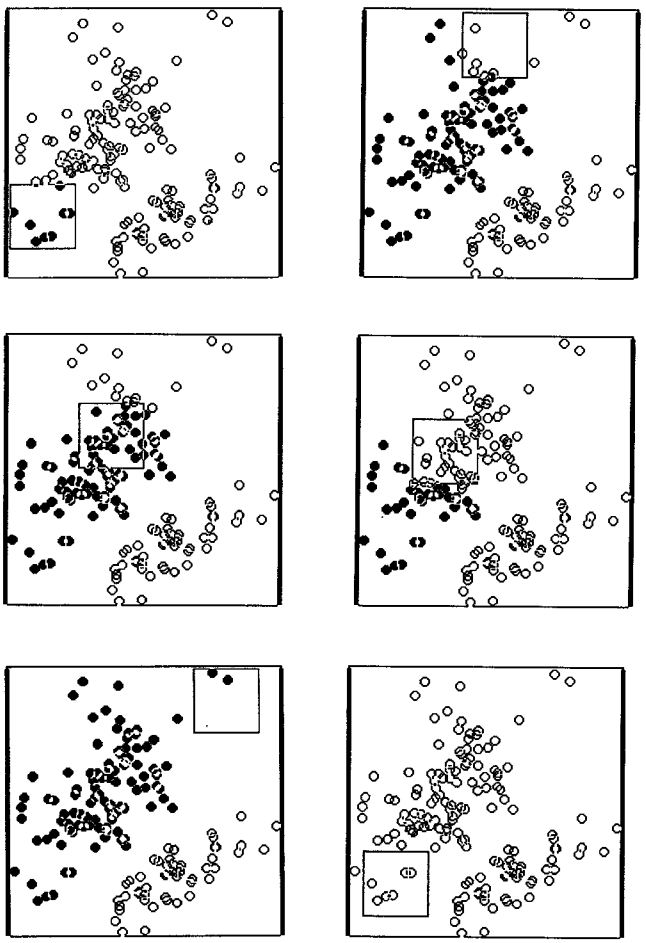

FIG. 10. *Lasting and undo paint modes. When the paint mode is lasting, points remain highlighted. This enables highlighting of irregular clusters by painting, as illustrated by the sequence of panels in the first column of the figure. When the paint mode is undo, lasting highlight is removed. This enables removal of lasting highlight by painting, as illustrated by the sequence of panels in the second column of the figure.*

Conditioning on Two Variables and the Shadow Highlight Operation. Figure 14 shows a scatterplot matrix with four variables: wind speed, temperature, solar radiation, and the cube roots of concentrations of the air pollutant, ozone. The original data, from a study of the dependence of ozone on meteorological conditions (Bruntz, Cleveland, Kleiner and Warner, 1974), were measurements of the four variables on 111 days from May to September of 1973 at sites in the New York City metropolitian region. There was one measurement of each variable on each day so the data consist of 111 points in a four-dimensional space. The analysis began with the cube root transformation of ozone because of the success of power transformations in previous analyses of air pollution data. Then a nonparametric regression curve was fit using locally weighted multiple regression, a local fitting procedure (Cleveland and Devlin, 1986); that is, cube root ozone was smoothed as a function of the meteorological variables. The data of the scatterplot matrix in Figure 14 are the three meteorological variables and the fitted, or smoothed, values of cube root ozone. Brushing was applied to these data to explore the fitted regression function, and to determine if there is appreciable nonlinearity in the surface; whether the nonlinearity exists is not immediately clear from the scatterplot matrix alone.

In Figure 15 highlighting is being carried out, but in a different way from that for the abrasion loss data that were analyzed in the previous section. First, although all data appear on the active panel, a scatterplot of radiation against temperature, only points that are highlighted appear on the other panels. This *shadow highlight* operation is useful in situations, such as this one, where the amount of overlap of the plotting symbols is so great that the highlighted symbols are hard to distinguish from those not highlighted. A second difference is that a square brush is being used in the active panel; thus, we are conditioning on values of radiation and temperature fixed to certain ranges. The points of the (1,4) panel show us how the smoothed cube root ozone values depend on wind speed for the other two independent variables held fixed to these ranges. This two-variable conditioning can be done anywhere in the (2,3) panel to condition on other ranges of radiation and temperature, or can be done in the (2,4) panel or the (3,4) panel to condition on radiation and wind speed or temperature and wind speed.

When brushing is used in this way, conditioning on two variables, it becomes quite clear that the fitted surface is highly nonlinear for the ozone data. For example, in Figure 15, where we are conditioning on middle values of temperature and high values of radiation, the dependence of ozone on wind speed has substantial curvature.

Brushing and Alternagraphics. Brushing in highlight or shadow highlight mode can also be used to perform alternagraphics. A categorical variable is defined that describes the subsets—for example, it takes on the value 1 if a point is in subset 1, 2 if it is in subset 2, and so on—and is plotted along with the other variables. Then, a highlighting of the different subsets can be gotten by applying conditioning to the categorical variable. It is also helpful to jitter the values of the categorical variable—that is, add small amounts of random noise—to reduce overplotting. This method is illustrated in Figures 16 and 17 for the brain and body weights discussed earlier. Figure 16 shows the full scatterplot matrix of the three variables: log brain weight, log body weight, and a categorical variable with jitter added that describes the species

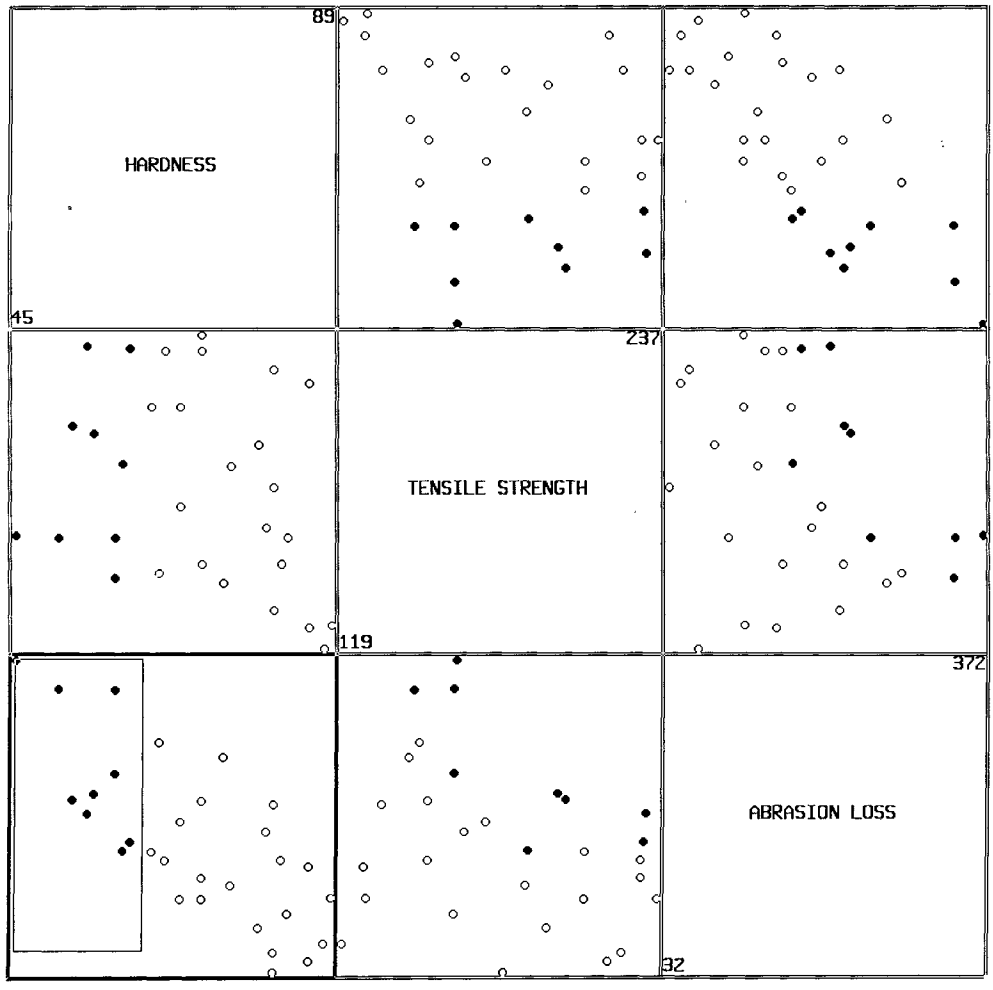

FIG. 11. *Conditioning. The brush is highlighting points with low values of hardness. The* (3,2) *panel shows the dependence of abrasion loss on tensile strength for hardness held fixed to low values.*

class. (From left to right, the classes are birds, fish, primates, nonprimate mammals, and dinosaurs.) In Figure 17 the mode is shadow highlight, and the brush in the (1,3) panel is selecting the value of the categorical variable corresponding to nonprimate mammals, so the scatterplot in the (1,2) panel is the brain and body weights for this group of species. By moving the brush on the (1,3) panel we can quickly select any other group.

The Delete Operation. The *delete* operation is carried out in the same way as highlighting. With no buttons depressed, points inside the brush on the active panel are deleted in a transient way along with corresponding points on other panels. Pressing a mouse button causes the deletion to be lasting, and pressing another mouse button causes the deletion to be undone, that is, deleted points reappear. Figures 18 and 19 illustrate the use of brush deletion in the analysis of a time series. The figures are based on the residuals, $x(t)$, of monthly atmospheric concentrations of CO_2 (Keeling, Bacastow and Whort, 1982) after trend and seasonal components were removed. The four variables in the scatterplot matrix are $x(t)$, $x(t-1)$, $x(t-2)$ and t. The (1,2) and (1,3) panels of Figure 18 suggest there are nonzero autocorrelations at lags 1 and 2. However, the (1,3) panel shows a peculiar excursion of the data near the end of the observation period. It is natural to ask what effect this excursion has on the autocorrelation. In Figure 19 a brush has been positioned in the (1,4) panel so that the points of the excursion are deleted. The autocorrelation at lag 1 is reduced and the autocorrelation at lag 2 disappears altogether; thus, much of the observed autocorrelation at these lags is induced by the data of the short, final time interval.

The Label Operation. The *label* operation is carried out in the same way as highlighting and deleting, that is, labeling can be transient, lasting, or undone. Figure 1, discussed at the beginning of the paper, illustrates the use of transient labeling.

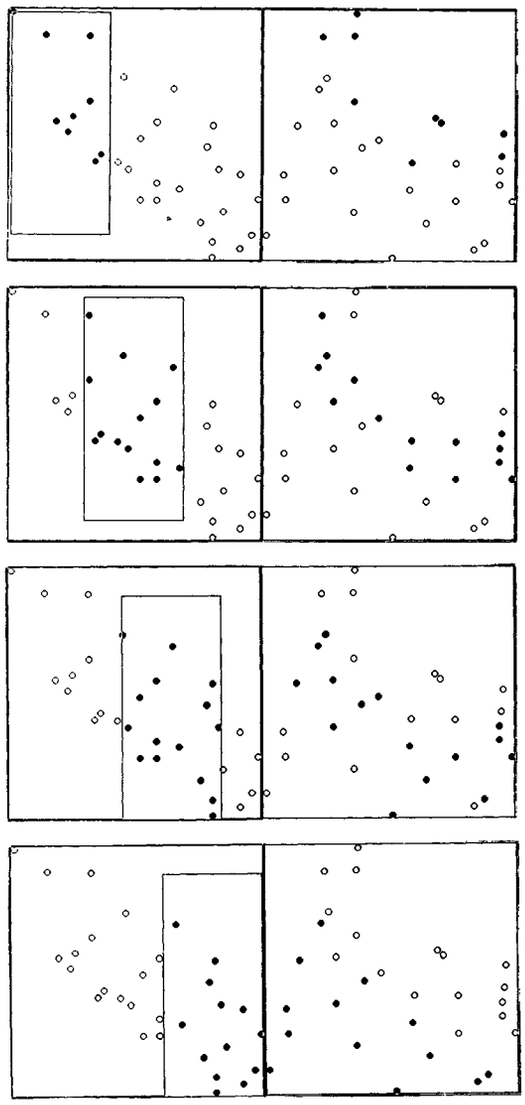

FIG. 12. *Conditioning. Each of the four pairs of graphs are the* (3,1) *and* (3,2) *panels of the scatterplot matrix in the previous figure. The brush is being moved from left to right on the* (3,1) *panel, conditioning on different ranges of hardness.*

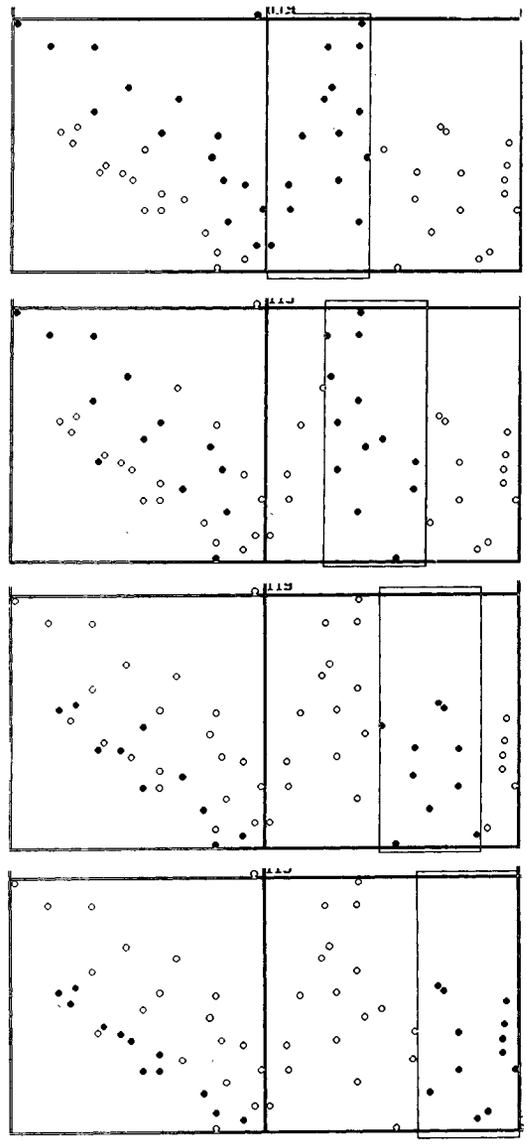

FIG. 13. *Conditioning. In this figure the brush is being moved on the* (3,2) *panel, conditioning on different ranges of tensile strength.*

2.5 Scaling

An important factor on a two-variable graph is the aspect ratio: the physical length of the vertical axis divided by the length of the horizontal axis. Changing the aspect ratio of a graph can have a radical effect on our ability to visually decode the quantitative information on the graph. Figures 20 and 21 are an example. The data are the yearly sunspot numbers from 1749 to 1924 that Yule (1927) analyzed in his landmark paper on autoregressive processes. In Figure 20 the aspect ratio is one and in Figure 21 it is 0.065. In Figure 20 we cannot see an important feature of the sunspot data that is apparent in Figure 21—the sunspot numbers rise more rapidly than they fall; that is, the absolute rate of change is greater when sunspot numbers are increasing than when they are decreasing.

In studying a two-variable graph, we often judge the slopes of line segments to determine the rate of change of y as a function of x. For example, in Figure 21 we judge the slopes of line segments connecting *successive* data points to determine that the sunspots rise more rapidly than they fall. In general the judgment of slope is enhanced when the angles of line segments with positive slopes are centered on 45° and when the angles of line segments with negative slopes are centered on −45° (Cleveland and McGill, 1987). When slopes tend away from ±45°, either toward the vertical or horizontal, relative judgments of slope decrease in accuracy. Changing the aspect ratio of a graph changes

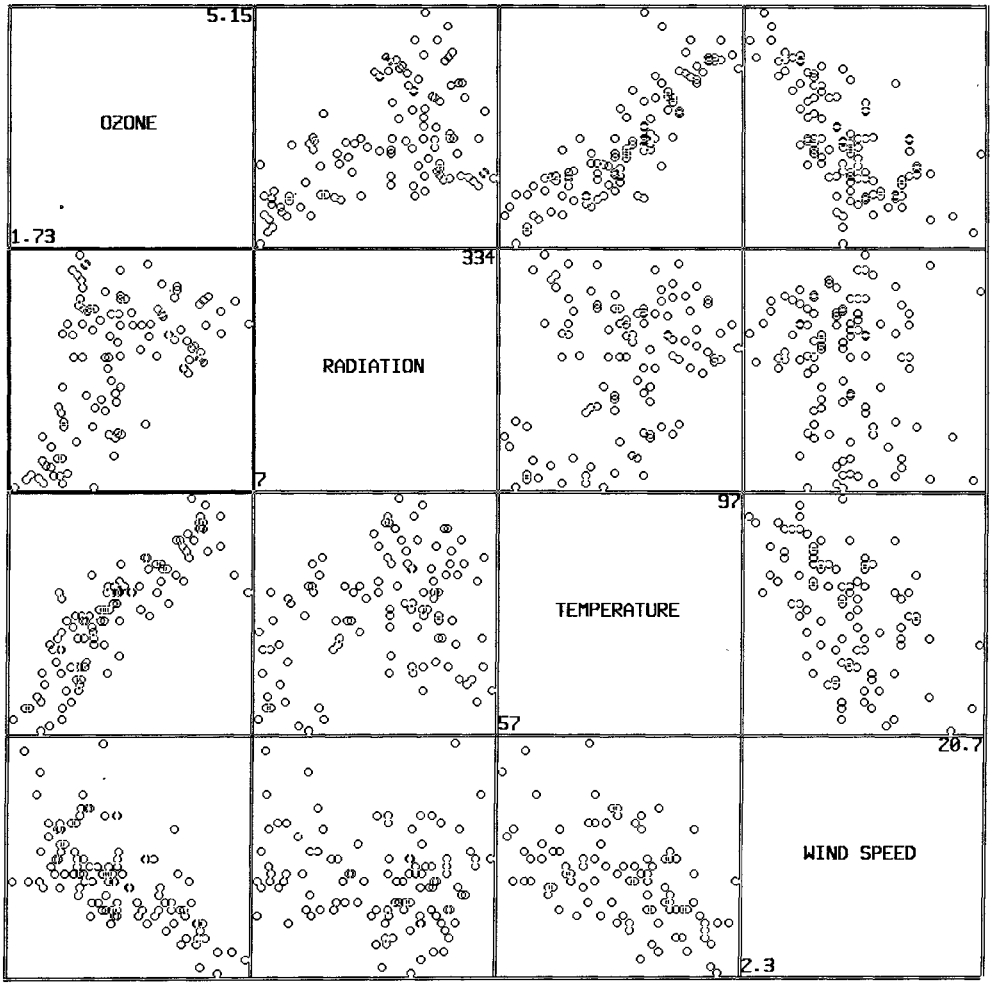

FIG. 14. *Scatterplot matrix. Four variables are shown.*

the angles of line segments on the graph and therefore our ability to judge slopes. In Figure 20 we lose our ability to judge local slopes because the angles of the line segments connecting successive points are too close to ±90°.

For static graphs the aspect ratio can be chosen by selecting a set of line segments whose slopes are of interest and then selecting the aspect ratio so that the median of the absolute values of the angles of the segments is 45° (Cleveland and McGill, 1987). But there are problems with this solution. One is that this optimum aspect ratio for one set of slopes might be quite different from that for another set of interesting slopes on the graph. Furthermore, optimum aspect ratios for many graphs, particularly those of long time series, are so small that for the usual page width (about 18 cm in this journal), the optimally scaled graph is too small in the vertical direction to have sufficient resolution along the vertical axis.

Dynamic scaling, in which the data analyst controls the length and width of the two axes, provides a convenient solution to these problems. The data analyst controls the aspect ratio and the computer makes the graph on the screen vary in a seemingly continuous way; the rapid and easy change from one aspect ratio to another allows accurate judgments of different sets of slopes. One particularly nice implementation of Buja, Asimov, Hurley and McDonald (1987) even solves the problem of small aspect ratios interfering with vertical resolution by using horizontal scrolling. To get a small aspect ratio without ruining the vertical resolution, the width of the graph is expanded beyond the width of the screen; the analyst sees only a portion of the graph at any moment and can scroll back and forth, bringing different parts of the graph into view in order to see all of it.

2.6 Rotation

Suppose x_i, y_i and z_i, for $i = 1$ to n, are measurements of three variables. The data configuration is a point cloud: n points in three dimensions. The computer

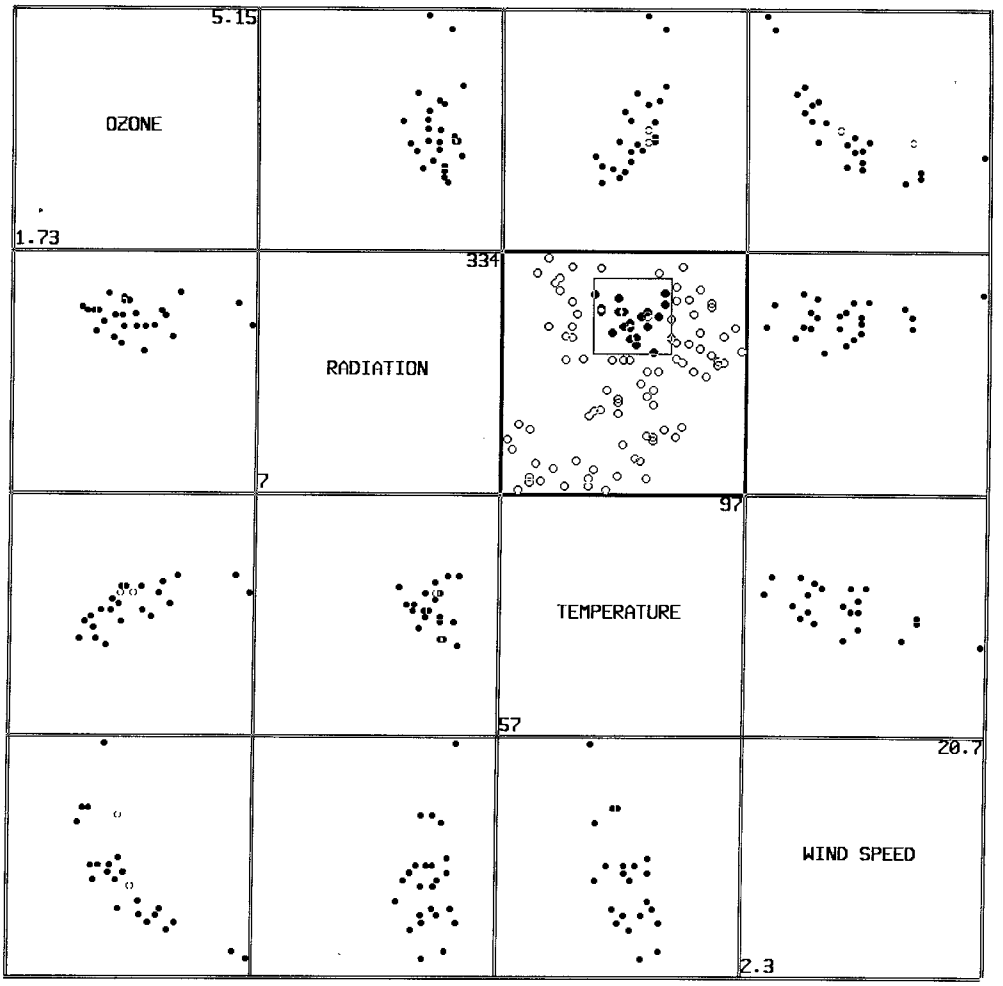

FIG. 15. *Conditioning with shadow highlight. When the brush operation is shadow highlight, all points on the active panel are shown, but on the other panels only highlighted points are shown. The points of the* (1,4) *panel show how ozone depends on wind speed for high values of radiation and middle values of temperature.*

screen is two-dimensional, so we must be satisfied with inferring the three-dimensional structure from two-dimensional views. This, however, is not as limiting as it might seem because one of the wonders of the human visual system is that its algorithms for perceiving the three-dimensional world have as their input two-dimensional views falling on the retinas. Just as a motion picture gives a three-dimensional sense from pictures shown on a flat screen, so can a computer display convey three-dimensional point clouds by a rapidly changing sequence of two-dimensional views (Fisherkeller, Friedman and Tukey, 1975). The point cloud is projected onto the screen while the viewing angle changes through time; the effect is an apparent rotation of the cloud that allows us to see its three-dimensional structure. Rotation is portrayed in Figure 22.

Control. There are many possibilities for selecting an axis about which the cloud rotates (Fisherkeller, Friedman and Tukey, 1975; McDonald, 1982; Tukey, 1987a). One is to use one of the three coordinate axes of the data. Rotation about one of them changes the orientation of the others from our fixed viewpoint—that is, the projection plane of the screen—so the apparent motion of the cloud about a given coordinate axis will vary according to the position of the other two axes. A second possibility for axis selection is to have three fixed screen axes, that is, axes that are fixed relative to the projection plane; the most sensible are screen horizontal, screen vertical and perpendicular to the screen. Once one of these six axes has been selected, the data analyst can be given control of the sign and speed of rotation.

A third possibility for axis selection can lead to very delicate control of the cloud. The data analyst selects a particular direction in the projection plane by moving a mouse or a tracker ball. The motion of the cloud is what we would get if we reached out and spun the

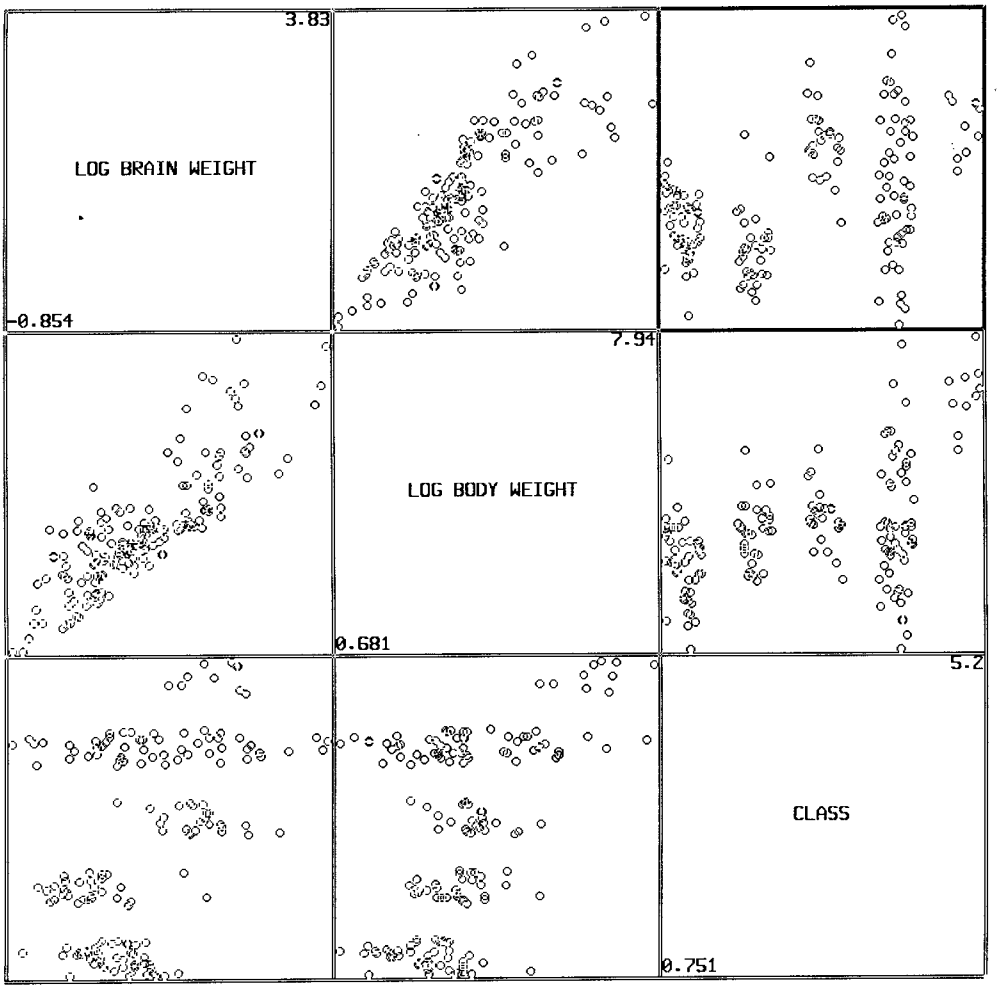

FIG. 16. *Alternagraphics by brushing. Brushing can be used to show different data sets by adding a categorical variable that indicates to which set each observation belongs. This figure shows a scatterplot matrix for the brain and body weight data, including a categorical variable identifying each species as belonging to one of five categories of animals. From left to right on the* (1,3) *and* (2,3) *panels the categories are birds, fish, primates, nonprimate mammals and dinosaurs.*

cloud by pushing on the front of it in in the selected direction. If the amount of rotation depends on how far the device is moved, the data analyst has the feeling of very direct control of the cloud. The cloud can also be given inertia, so that once it is set in motion it continues until the analyst changes the motion.

It is entirely reasonable to implement all three of the above rotation-control methods in a dynamic graphics system and allow each data analyst using the system to discover his most comfortable control method.

Enhancements. Motion is only one factor that the human visual system uses to construct scenes; consequently, rotation by itself does not give as strong a three-dimensional view as we get of the real world. Remember that points in three-space are not typical of the real world—real objects are composed of surfaces, not points. However, certain enhancements to rotation can increase the perception of depth. First, we can add stereo vision (Becker, Cleveland and Weil, 1987; Donoho, Huber, Ramos and Thoma, 1986; Littlefield, 1984). One way to do this is to show two superimposed views of the cloud from two slightly different viewpoints, one view in red, the other in green, with overlap in yellow, and then view the display with glasses that have one red lens and one green. Each eye gets one view as in real-world vision. (There are other methods of generating stereo views that allow color in the pictures.)

Stereo vision enhances the three-dimensional effect of the rotating cloud; but, even more importantly, the three-dimensional effect remains even when the motion stops. This is important for reasons that will be given shortly. Because our visual systems also use perspective to see depth, we can enhance point cloud rotation by having the sizes or intensities of

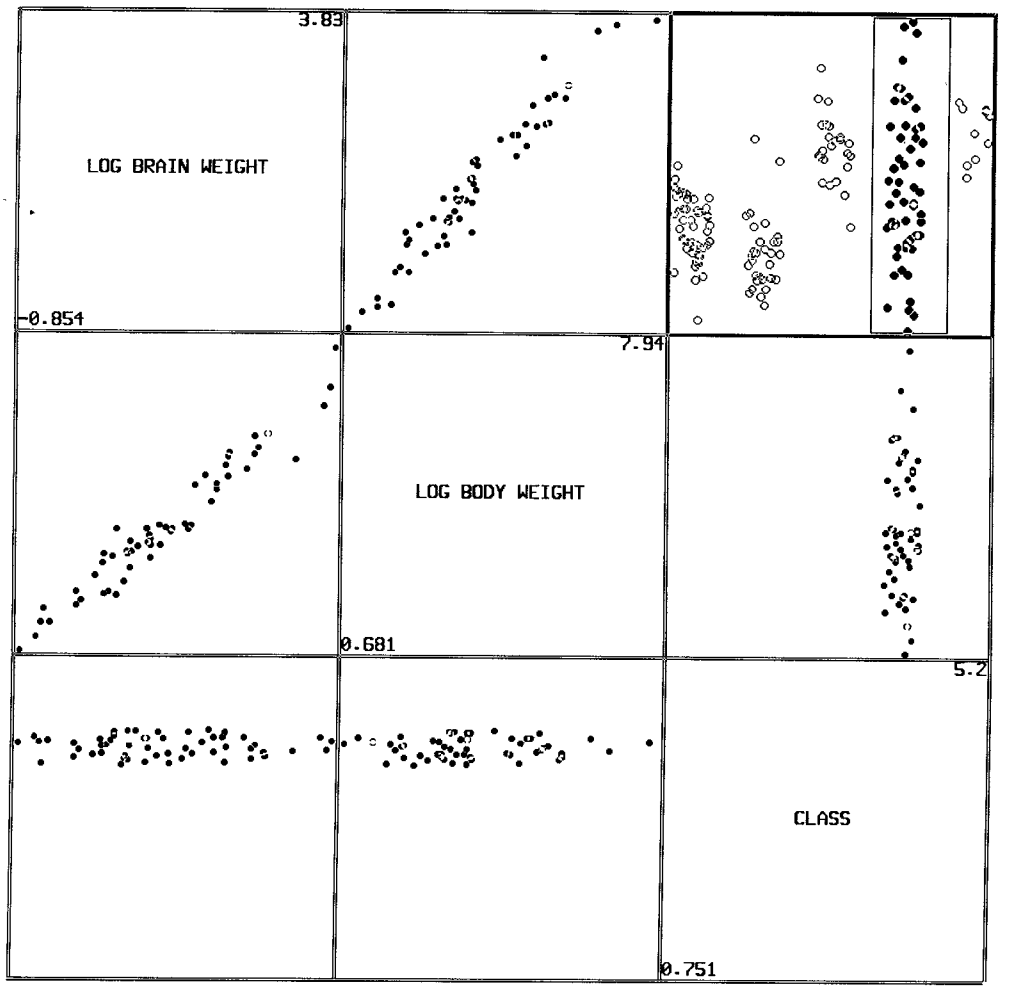

FIG. 17. *Alternagraphics by brushing. Here one of categories (nonprimate mammals) of Figure 16 has been highlighted in shadow highlight mode. By choosing a tall narrow brush and moving it across the (1,3) panel, each category is momentarily visible on all of the plots.*

the plotting symbols obey the rules for perspective. Another way to enhance the three-dimensional effect is to enclose the cloud in a rectangular box whose edges are the axes of the three variables; the box provides perspective, which enhances the depth effect, and also helps us perceive the axis directions.

Limitations. It should not be supposed that rotation and its enhancements give us for three variables exactly what scatterplots give us for two. On a scatterplot, we make a variety of judgments of horizontal and vertical positions to visually decode information (Cleveland, 1985). Point cloud rotation and its enhancements add depth, but our judgment of a position in depth is not nearly as acute as our judgment of horizontal and vertical positions. Furthermore, when a point cloud is rotating it is easy to lose a strong cognitive fix on the axes and on which variable goes with which axis, even when the axes are shown. It is almost always the case in analyzing a set of data that recognizing some pattern in the data is not enough; we need to be able to interpret the pattern in terms of the measured variables. This is illustrated in the (1,2) panel of Figure 6. We can readily perceive from this scatterplot that there are two clusters of points, and we can almost instantaneously conclude that the cluster in the upper left consists of sepals that are shorter and wider than those in the lower right. For a scatterplot this process of interpreting a perceived pattern in terms of the measured variables is done so effortlessly that we are hardly aware that it is distinct from perceiving the pattern. For a rotating three-dimensional point cloud we are often painfully aware of the separation. The problem is alleviated somewhat by having stereo; then the motion can be stopped and a particular view contemplated. But even with this enhancement, our cognitive processes are considerably less adept than they are for two dimensions.

Rotation with More Than Three Variables: Advanced Strategies. Suppose there are p variables

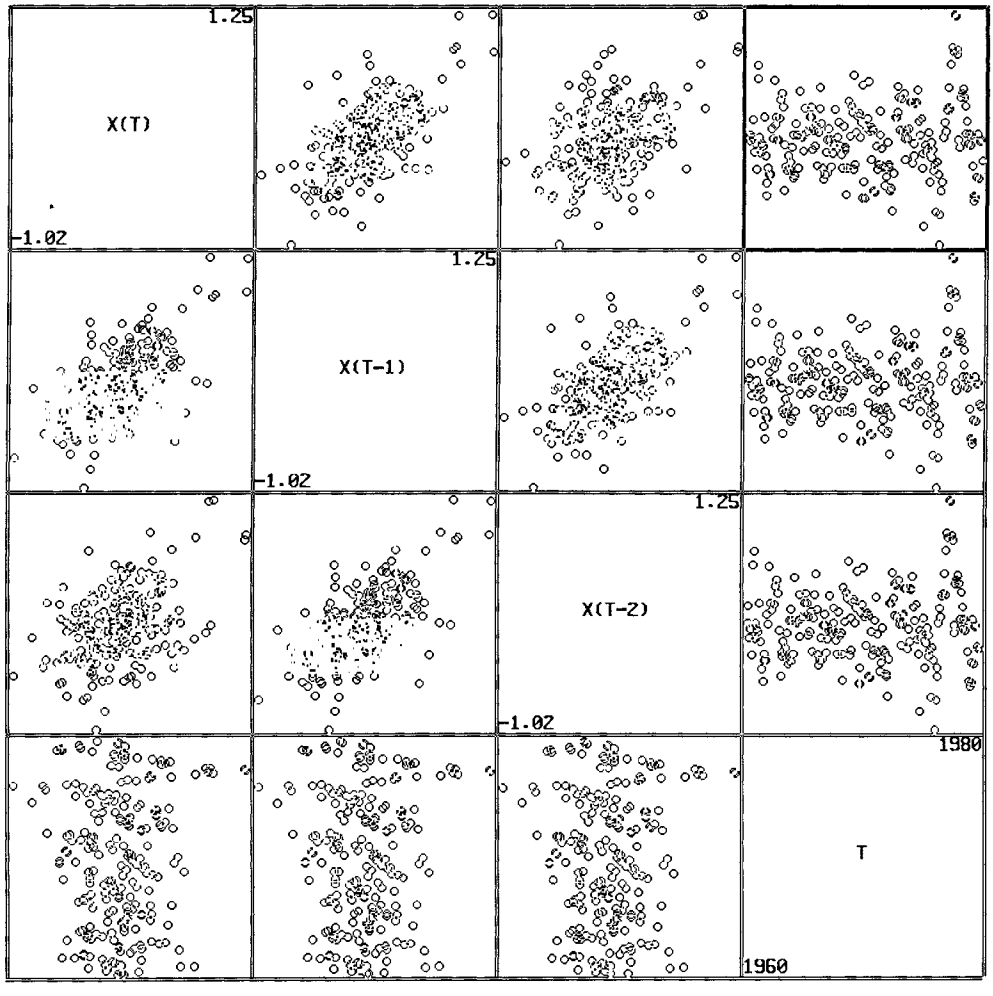

FIG. 18. *Time series. A time series, $x(t)$, is explored by the scatterplot matrix. The* (1,3) *and* (2,3) *panels suggest that there is significant correlation at lags* 1 *and* 2. *However, panel* (1,4) *shows there is a peculiar excursion of the data over a short time interval toward the end of the data record.*

where $p > 3$. We can study any three at a time by rotation. If we do this, without saving the results of a rotation in any way, we can effect rotation only about axes that are a linear combination of three variables. We can, however, adopt a replacement strategy, similar to one used in PRIM-9 (Fisherkeller, Friedman and Tukey, 1975), that allows arbitrary rotations. Think of the procedure as a recursion with a succession of rotation sessions. After the kth session, suppose the data are described by a coordinate system with variables $z_1, \ldots, z_p$. Three of the variables, say z_1 to z_3, are selected for the next session. The data are rotated to get a final view for the session. The variables z_1, z_2 and z_3 are now replaced by a new set of three variables: the three that describe the screen axes. At the beginning of the sessions, the z_i are just the original data variables. Needless to say we rapidly lose a graphical appreciation of the meaning of identified structures in terms of the original variables. However, the system can keep track of the coefficients of the linear transformation of the z_i back to the original variables as way of helping us to infer meaning.

During these rotation sessions we can also condition on certain ranges of the z_i (Donoho, Donoho and Gasko, 1985; Fisherkeller, Friedman and Tukey, 1975). For example, we might rotate with z_1, z_2 and z_3 and show only points for which $a < z_4 < b$, which is a form of deletion, or we might highlight points in this range. This allows us to study the relationship of z_1, z_2 and z_3 conditional on z_4 in the range a to b. In addition, we can hold the rotation fixed to a particular view and vary a and b.

Far more experimentation is needed with these advanced strategies. In applications, it is easy to lose all sense of where one is and what identified structures mean. Carefully worked-out examples, complete with strategies and what they show about the phenomenon under investigation, are needed.

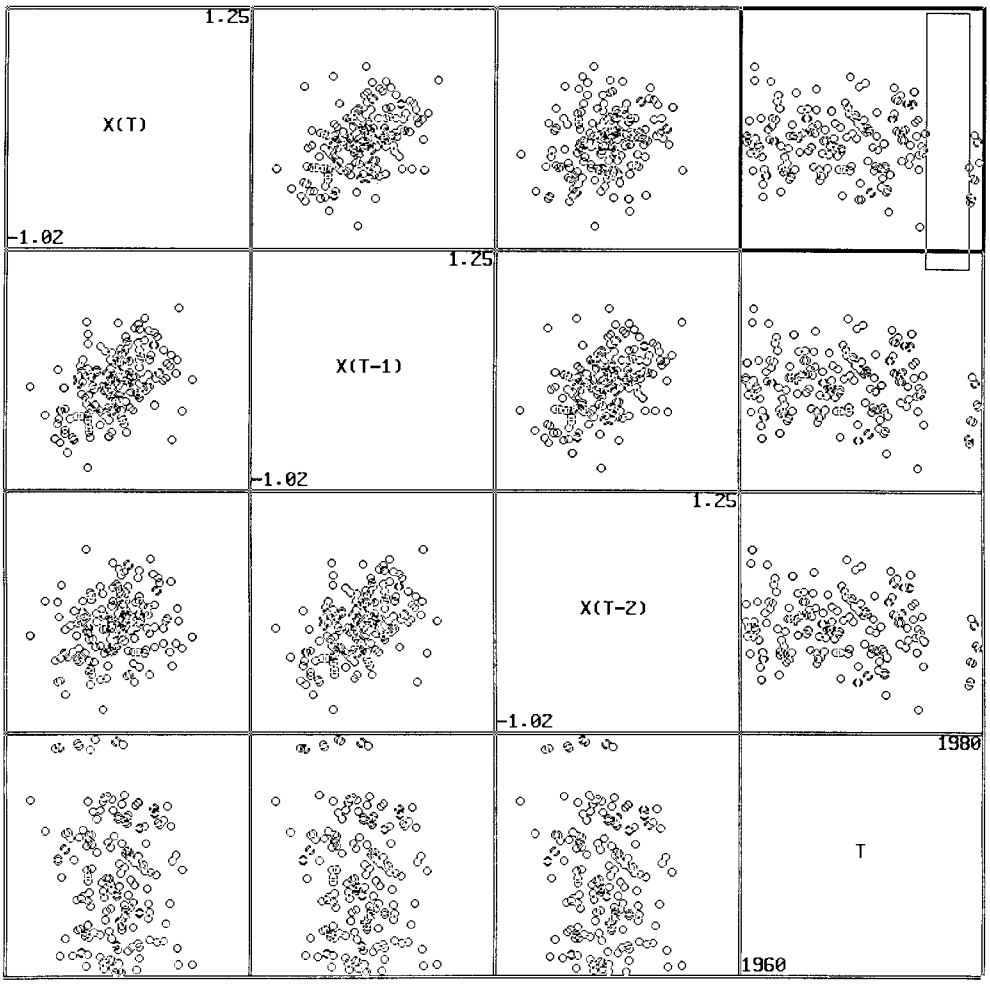

FIG. 19. *Deletion by brushing. Points can be deleted with brushing in the transient or lasting paint mode, and lasting deletion can be undone by the undo mode. In this figure, the brush is removing the data of the short excursion. The autocorrelation at lag 2 has disappeared and the autocorrelation at lag 1 has been substantially reduced.*

Brushing vs. Rotation. Brushing and rotation can both be used for studying multidimensional data. Becker, Cleveland and Weil (1987) compared the two methods by implementing both on a high performance graphics workstation and using both to analyze a large number of data sets. We will not repeat the details of the investigation here, but will describe the basic conclusions.

Suppose the data are three-dimensional and the goal is to examine the trivariate structure. Simply because we can see directly the three-dimensional structure, rotation is particularly useful for exploring three-dimensional properties such as clusters and collections of points that lie close to one-dimensional and two-dimensional manifolds. Brushing tends to be less useful here, except in special cases.

Suppose the data are three-dimensional and the goal is to explore the local dependence of one variable on the two others. Again, rotation is a useful tool and brushing tends to be of less help.

Suppose the data are three-dimensional and the goal is to explore the conditional dependence of one variable on another for the third held fixed. Here brushing is particularly appropriate and rotation tends to perform less well. The reason is that the conditioning techniques of brushing allows us to incisively study conditional dependence, the usual goal, for example, when we do a regression analysis.

Once the data are in four dimensions or more the picture becomes somewhat less clear. If the goal is to study dependence of one variable on the others, brushing typically leads to more insight. For multivariate studies where understanding the p-dimensional structure is the goal, both methods can lead to separate and important insights. For example, when a cluster is definable by its projection onto the plane of two original variables, it can be explored through highlighting the cluster by brushing the scatterplot of the two variables. However, if the cluster is definable by a projection onto the space of three original variables,

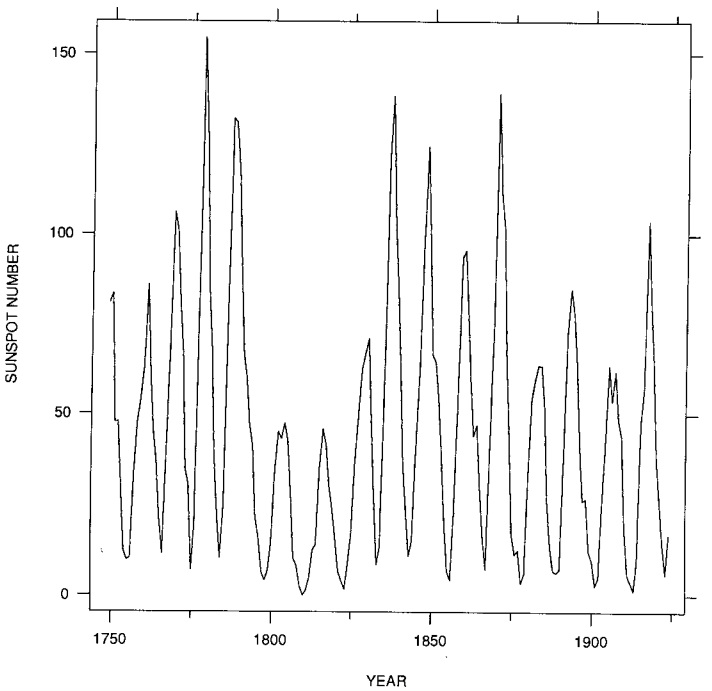

FIG. 20. *Dynamic shape control. The yearly sunspot numbers are graphed. The shape parameter of the graph—the vertical distance spanned by the data divided by the horizontal distance spanned by the data—is equal to one. From this graph we get a good portrayal of the variation in the peaks of the sunspot numbers but a poor portrayal of the shapes of the cycles.*

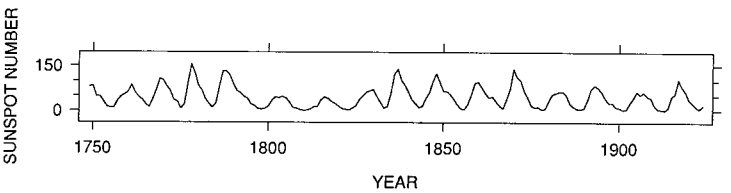

FIG. 21. *Dynamic shape control. The shape parameter of the graph is now* 0.065 *and we get a good portrayal of the shapes of the cycles— for example, we can see that the data tend to rise more rapidly than they fall—but a poor portrayal of peak to peak variation. Dynamic graphics can be used to continuously vary the shape parameter of a graph, quickly moving from this figure to the previous one.*

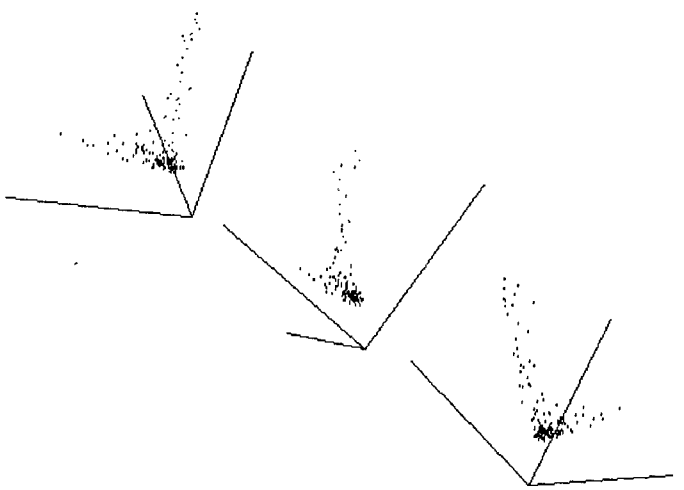

FIG. 22. *Rotation. Rotation on a computer graphics screen of measurements of three variables allows us to see the point cloud in depth, which provides a three-dimensional scatterplot. Rotation is a particularly effective method when we are searching for trivariate structure, such as clusters or manifolds, or when we want to study the local dependence of one variable on the other two. When the goal is to study conditional dependence, brushing is typically more effective. The portrayal of a rotating point cloud in this figure is from the MACSPIN manual (Donoho, Donoho and Gasko,* 1985*). Reprinted with permission of* D^2 *Software, Inc.*

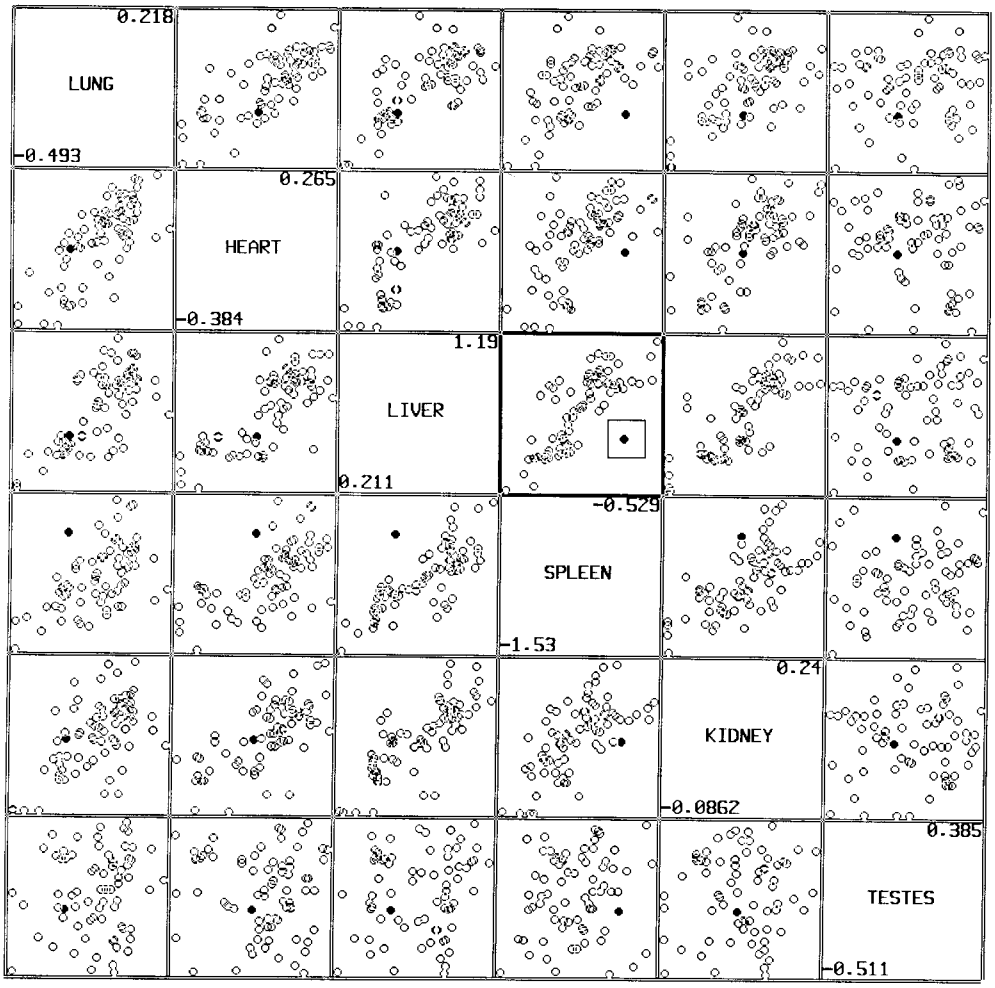

FIG. 23. *Four or more variables. It is quite sensible to use both brushing and rotation when there are four or more variables. One reason that brushing is effective is the single integrated view provided by the scatterplot matrix. In this figure the logarithms of the weights of hamster organs are graphed. There is an outlier in the* (3,4) *panel, which is being highlighted by the brush. By scanning rows* 3 *and* 4, *we can see the hamster has an enlarged spleen.*

but not onto any plane of two of the variables, it is detectable only through rotation.

One strength of brushing for four or more variables is that the scatterplot matrix provides a single integrated view of the data unless, of course, a large value of n or p causes resolution problems. Even when several rotating clouds are on the screen, the same level of integration in not achieved. The integration allows a number of data analytic tasks to be carried out in a straightforward manner. An example is provided by the data on Figure 23, which are the logarithms of the weights of six body organs for 73 cardiomyopathic hamsters (Ottenweller, Tapp and Natelson, 1986). The outlier in the (3,4) panel, which is being highlighted by the brush, suggests that one hamster has an unusually large spleen or an unusually small liver. By scanning the spleen plots in row 4 and the liver plots in row 3, we can see that the hamster has an enlarged spleen.

Thus, the overall conclusion is that neither brushing nor rotation uniformly dominates the other and it is quite reasonable to have both tools available. In fact, not only do brushing and rotation complement one another, they can be combined to create more than the sum of their parts (Becker, Cleveland and Weil, 1987; Stuetzle, 1987). A subset of three of the variables is selected and rotation is applied to them on a panel adjacent to the scatterplot matrix. This panel takes part in the brushing operations just like the panels of the scatterplot matrix. When a matrix panel is brushed, the results are also shown on the rotation panel; furthermore the rotation can be stopped at a particular view and the rotation panel selected as the master panel for the brushing operations.

2.7 The General Case: Parameter Control

Many of the methods described in the previous sections provide the data analyst with a method for

rapidly changing a parameter that affects a graphical display. The following are examples: alternagraphics—the value of the categorical variable that breaks the data up into subsets; scaling—the height or width of the graph; rotation—angle of view. Dynamic methodology can in principle be used to control any parameter, discrete or continuous, that can affect a graphical display. Two further examples are the powers of power transformations of variables on a graph (Becker and McGill, 1985; Fowlkes, 1971) and the smoothness parameter of a nonparametric regression curve superimposed on a two-variable graph (Friedman, McDonald and Stuetzle, 1982). Thus, there are a vast number of methods that can potentially be implemented in a dynamic graphics environment.

3. COMPUTING ISSUES

This section focuses on some of the computing issues involved in *developing* a dynamic method. The discussion is needed because of two circumstances: current hardware must be pushed nearly to its limits to achieve the requisite speed for dynamic methods, and present software environments leave the programmer with many steps between the conception of a new idea and an easy to use implementation of it. With present technology, developing easy to use, responsive methods requires some knowledge of hardware and software issues.

Two aspects of the hardware for a graphics computing system have a large impact on the speed and ease of use of dynamic techniques: the bandwidth of the complete system and the specific input/output devices used in the system. These are discussed in Sections 3.1 and 3.2.

There are several levels of software used in the implementation of dynamic displays—low level software that operates on bits in frame buffers, intermediate level software that provides basic graphics library functions and high level software that provides an *environment* where data analysts can make good use of dynamic graphics. The levels of software needed to generate a particular dynamic display are determined by many factors including the complexity of the display and the hardware on which it will run. A simple dynamic method may be trivial to implement on very powerful hardware. Conversely, a graphical method requiring fast display of large amounts of data on an inexpensive computer system will probably force the programmer to use many low level graphics algorithms in order to get adequate performance.

Low level graphics techniques are described in Section 3.3. These are the techniques typically used by programmers who need maximal control over the graphics display because higher level software either does not perform the right task or does not perform the task with enough speed. We then discuss higher level, device-independent techniques in Section 3.4 are easy to use and are accessible to programmers and to data analysts with programming experience. Finally, in Section 3.5, we take up the data analysis environments that make dynamic graphics displays available to data analysts.

3.1 System Bandwidth

Bandwidth is the speed with which user input, such as mouse moves and button presses, can be translated into picture updates on the screen. There are several steps in this translation, any of which can be the bottleneck that determines the overall bandwidth. Composite speed may be limited by the speed of numerical or graphics computations, by the speed of the display hardware or by the speed of passing information from input device to computer or from computer to display device. It is hard to quantify the required overall speed, but as a general rule of thumb, the bandwidth should be sufficiently high that the *perceived* response of the display to the input device is virtually instantaneous. McDonald and Pedersen (1985a) attempt to quantify hardware bandwidths of the various components of modern workstations, and also provide a general background on hardware from a data analyst's viewpoint.

Roughly speaking, the hardware necessary for doing dynamic graphics consists of computing horsepower sandwiched between input and output devices. The ways in which such a complete system is typically arranged can be broadly characterized as either the *time-shared* model, the *workstation* model or the *distributed processing* model, which is often a hybrid of the other two. Each organization has implications for dynamic graphics.

In the time-shared model (Figure 24a), the user has a terminal that is attached to a host computer. The terminal's display and input device are controlled by a fixed *terminal program* that runs in a processor inside the terminal. Because the terminal program is fixed, graphics capabilities will be controlled from the host computer by relatively primitive commands. This means that dynamic graphics techniques are limited in this model by the amount of information that can be exchanged between the host and the terminal; typical host-terminal communication speeds are relatively low. For example, to display a thousand points on a terminal might typically require sending four bytes per point. The communication channel between terminal and host rarely transmits more than 2000 bytes per sec, so that each display of a thousand points would take a minimum of 2 sec to send; this is much too slow for dynamic graphics. Furthermore, although host computers in a time-shared environment may

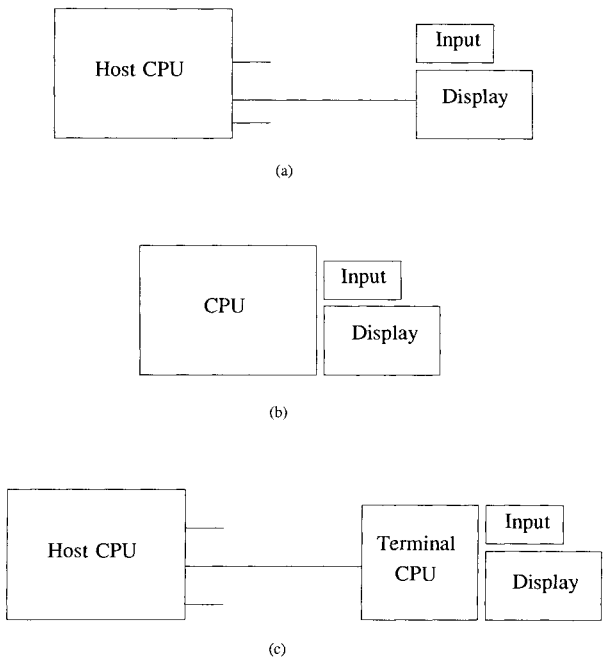

FIG. 24. *Models for graphics systems. Three models that are used for computing systems: (a) the time-shared model, in which a host computer is shared among several users, each with a terminal that has a fixed, relatively simple terminal program; (b) the workstation model, in which the user has the exclusive use of an entire computer (but perhaps sharing resources such as disk); and (c) the distributed computing model, in which the graphics computations are split between a shared host computer and a dedicated, programmable local computer.*

have a fast processor, the response time for a particular user can be slow because of other users employing the system. Because of these speed problems, the time-shared model is generally unsatisfactory for dynamic graphics.

In the workstation model (Figure 24b), the data analyst has the sole use of a computer. In order to be usable for dynamic graphics this computer should have, as a rough minimum, a processor that can perform one million instructions per second, a main memory capacity of one megabyte, a minimum screen resolution of about 500 by 500 and a high speed network connection to access a disk or other workstations—this connection should have a minimum speed of about one million bits per sec. These requirements are discussed in more detail by Morris, Satyanarayanan, Conner, Howard, Rosenthal and Smith (1986). The great advantage of the workstation model is that because the processor, the graphics display and the input device are all housed together, they can be connected by high speed communications channels. Also, because the computer is dedicated to one person, there are no time-sharing delays. Furthermore, workstations are becoming affordable for most data analysts.

The distributed processing model (Figure 24c) is often a hybrid of the time-shared and workstation models. The user has a programmable computer with a graphics display and this local computer is attached to a time-shared host computer. Examples of distributed configurations include the AT&T Teletype 5620 programmable graphics terminal and Sun Microsystems' NeWS window system. In this model, the graphics computations are carried out in several processors and the distribution of the computational tasks is under control of the software rather than the hardware. The information flow between the host computer and the local graphics computer may contain programs as well as data. Typically, extensive computations are carried out at the more powerful host machine and time-critical graphics operations are performed by the local graphics machine.

There are several advantages of the distributed processing model over the workstation model. One is that the local graphics computer can be totally dedicated to a particular graphics task, whereas a workstation will also typically have to perform other tasks associated with a sophisticated operating system. Because the cost of computing is shared among a number of users on a time-shared host, more computing power may be available to the person with the large host and local graphics processor than the the person with the workstation. This has an impact on any preprocessing that needs to be done for a dynamic technique. Also, the host machine gives the user the advantages associated with a shared environment, such as access to the files of others (although, with the advent of network file systems, this is also available in the workstation model).

The chief disadvantage of the distributed processor model is that software design is more complicated. The implementation requires part of the software to be running on the host and part to be running in the local machine; deciding how to do this partitioning can be delicate (Pike, Guibas and Ingalls, 1984).

3.2 Graphics Input and Output

Most current systems use a *mouse* for graphics input and a *raster scan display* for graphics output. A mouse is a small hand-held device that keeps track of relative changes in its position on a flat surface. These changes are continuously sent to the computer, which is then responsible for keeping track of the position of the mouse. The mouse is typically equipped with one, two or three buttons and the state of those buttons, pressed or not, is also continuously available to the computer.

A raster scan display has much in common with a television. It has an electron gun behind the screen that is continuously re-exciting spots on

the screen with short persistence, i.e. that quickly fade if not re-excited. The electron gun follows a fixed scanning pattern from left to right, top to bottom. The value for each spot on the screen (on/off or a color) is stored elsewhere in fast memory (called a *frame buffer*) and these values are read in synchrony with the scanning of the electron gun.

An asset of raster scan devices is their low cost. Raster scan hardware has two parts, the drawing hardware (screen, electron gun, etc.) and the associated frame buffer that describes the image. The drawing hardware is inexpensive because it is little different than television technology. Raster scan was once expensive because of the cost of the associated memory. Plummeting memory prices, however, have changed this. For example, a medium resolution color screen might have a 1024 by 1024 array of *pixels* (a pixel, or picture element, is a frame buffer location, corresponding to an individual spot on the screen) with 8 bits of information stored for each pixel, and this requires one megabyte of memory. In the past 5 years, the cost of memory has decreased 10-fold. This trend, more than any other, has accounted for the recent upsurge in the popularity and availability of raster scan devices.

Raster scan devices also have added flexibility in the frame buffer for color displays. In these displays, each screen pixel is associated with a frame buffer location where several (usually from 4 to 32) bits of information are stored. Most modern color displays use these bit patterns as indices into a *color map*: a table of color definitions. The arrangement of the frame buffer—several bits for each pixel—can be viewed alternatively by considering, say, the first bit for each pixel as a collection; this rectangular collection of bits is called the first *bitplane*, and the other bitplanes are similarly defined (see Figure 25). Most devices, under control of a numerical *read mask*, have the ability to display any subset of the bitplanes at any time.

Some hardware offers the ability to *pan* and *zoom*. When the display hardware is reading the frame buffer, it can be set up to begin its scan anywhere in the frame buffer—the visual effect is to shift the picture horizontally or vertically (or both) with wrap-around. In addition, the hardware can be instructed to replicate the values it reads from the frame buffer so that what would have been one pixel becomes a block of identical pixels on the screen. The visual effect is to zoom in on a part of the picture.

All of these features—color maps, bitplanes, zoom and pan—can be used separately or together to make rapid display changes, and are thus good grist for the dynamic graphics mill. There are some subtle tradeoffs involved between them, however. For a good discussion of the techniques and tradeoffs, see Heckbert (1984).

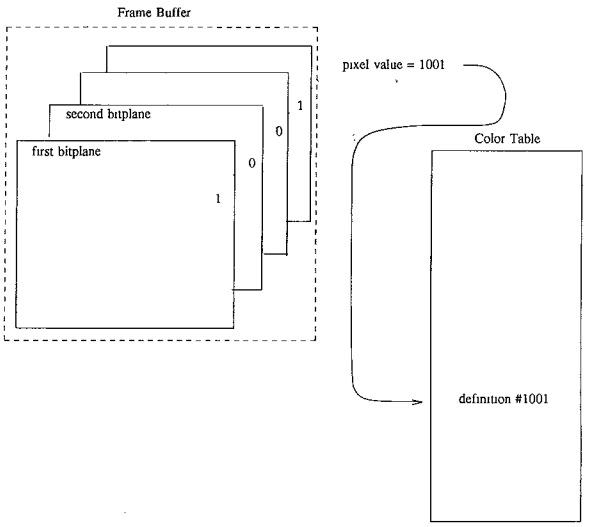

FIG. 25. *Color displays. Relationships between the frame buffer, bitplanes and color map in a typical color device are shown here. The frame buffer may be viewed as a stack of bitplanes, here four. Reading the bits in the same position for each plane gives the value of the screen pixel at that position. The pixel value is used as an index into a table of color definitions, each of which may be set independently of the values in the frame buffer.*

3.3 Low Level Graphics Algorithms

It is often necessary to customize software to particular hardware in order to get maximum performance from a graphics device. Performance is essential to the success of dynamic methods and often the hardware used for dynamic graphics is just barely capable of achieving adequate performance levels. Even with very powerful hardware, it usually takes little time before data analysts push the limits of the hardware by wanting to increase the number of points that are displayed, to add more complex numerical methods to the display computations or to complicate the methodology in other ways. It is easy to push even the most powerful hardware to its limit. The remainder of this section describes some of the low level algorithms useful for squeezing graphics performance out of various types of hardware.

Integer Arithmetic. One technique applicable to many display devices is the use of integer arithmetic for time-critical computations. This is especially useful on devices without floating point hardware that would otherwise need to use (slow) software floating point routines. Another advantage of integer arithmetic is that it tends to match the resolution available on graphics devices. Display devices often have vertical and horizontal resolutions of fewer than 1000 pixels. This means that computations accurate

to 10 bits are often sufficient for graphics; these can be done with hardware supported 16-bit integer computations. This integer arithmetic can be fast and can also save space. On the other hand, more programming care is required in translating ideas that are naturally expressed as floating point computations into equivalent integer computations.

Precomputation. Tradeoffs of speed versus memory usage are generally as applicable to graphics as they are to other computer algorithms. Time versus memory tradeoffs can be made by using precomputations; for example, sines and cosines can be precomputed and stored in a table so that trigonometric computations can be avoided during the display. Other tradeoffs involve precomputed pictures.

Suppose we would like to produce an alternagraphic display (Tukey, 1982). Sometimes, precomputing the pictures is the easiest method of accomplishing this; in this case, the length of time it takes to generate the pictures is then not linked to the speed at which they can be displayed. When the displays are precomputed and then shown, the procedure is usually referred to as an *animation sequence.* Such a sequence might be used to show the contours of a surface varying in time or to show successive slices through a three-dimensional set of data. In these cases the sequence can rapidly show (say from 3 to 30 pictures per sec) a series of pictures that change slightly from one to the next, giving the viewer the impression of a continuously varying display. Alternagraphic displays also cycle through a set of pictures, although typically more slowly (a second or so per picture), and in this case, the various pictures would usually change substantially from frame to frame.

One of the major advantages of using precomputed pictures is that there is no need to write special purpose software to create the pictures to be displayed. All that is necessary is to augment the available graphics software so that it can draw pictures in portions of selected bitplanes. Once the pictures are drawn, then it is a simple matter to write a program to display them alternately on the screen. The precomputed pictures can also be stored offscreen and then rapidly copied into the active memory when they are to be displayed. Later, we will discuss operations for copying bitmaps. More complicated schemes are possible on devices with color maps, bitplanes, zoom and pan (Heckbert, 1984).

An example of the use of precomputation for dynamic graphics involves producing a power transformation display. Suppose we have paired observations x_i and y_i for $i = 1, \ldots, n$. A power transformation can often be used to linearize a curved plot, to stabilize variance or to enhance symmetry. It is sometimes difficult to tell what will happen if we transform both variables simultaneously. Therefore, we want to allow two-dimensional motion of an input device to select transformation powers α and β and to show the changing scatterplot of y^β against x^α. In order to prevent scale changes from affecting the plot, we show $y_i^\beta/(\max(y)^\beta - \min(y)^\beta)$ against $x_i^\alpha/(\max(x)^\alpha - \min(x)^\alpha)$. We also need to make corrections for the sign of α and β as well as use a logarithmic transformation for $\alpha = 0$ or $\beta = 0$.

Suppose the hardware configuration is one in which computation is not enormously fast but there is a fair amount of memory. Straightforward computations would require floating point arithmetic to compute $\exp(\alpha \log(x_i))$ and $\exp(\beta \log(y_i))$ for each data point each time α and β changed. This is a lot of computation to do at a rate of approximately 10 times per sec, even for a moderate amount of data, say several hundred observations.

We could use tables of $\log(x_i)$ and $\log(y_i)$ to speed up this computation. However, by carrying precomputation just one step further, all floating point arithmetic can be avoided during display. Select a set of reasonable α values; for example, the 81 values $\alpha_j = -4$ (.1) 4. Next, compute a matrix with n rows and 81 columns where the i, j entry is $x_i^{\alpha_j}/(\max(x)^{\alpha_j} - \min(x)^{\alpha_j})$. Compute a similar matrix for y and β values. (We cannot compute just one table for grouped x and y because the denominators differ for x and y.) To display the transformed scatterplot, it is simple to find the column of the x matrix corresponding to the current value of α and the column of the y matrix corresponding to the current value of β. All that the device has to do is quickly produce the scatterplot from the precomputed values. Notice that it would probably be infeasible to precompute all possible 81^2 pictures in advance; instead we have stopped just short of producing pictures during the precomputation. The important feature of this problem that allows us to succeed is that the power transformation can be decomposed into separate operations on the x and y coordinates.

Raster-op. Recently, bitmapped display devices have become very popular. Most graphics on these devices is carried out by one very powerful operator, known as *raster-op* or *bitblt* (Newman and Sproull, 1979; Pike, Guibas and Ingalls, 1984). Many bitmapped display devices provide special hardware to assist raster-op.

The raster-op operator takes two rectangular arrays of bits, the *source* and the *destination*, and combines them using various functions. The function *store* copies the source rectangle to the destination (ignoring any bits that were present in the destination); the *or* function produces a one bit wherever a one bit appears in either the source or the destination; *exclusive-or* (XOR) produces a one bit whenever the source and destination bits are not the same. See Figure 26.

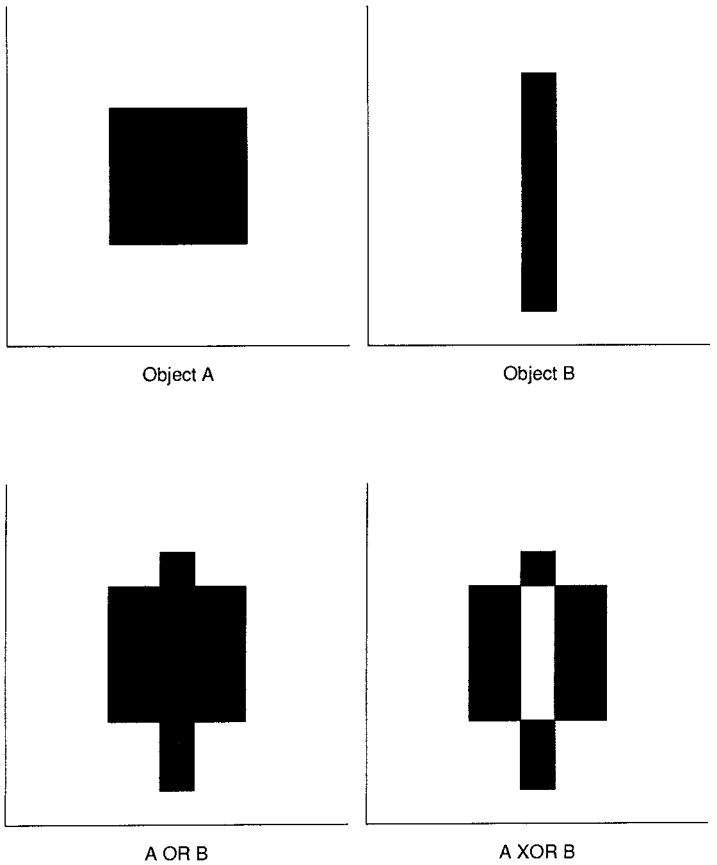

FIG. 26. *XOR graphics. The results of combining two objects using the OR and XOR raster-op operations. Black represents* 1*-bits and white represents* 0*-bits. Object A OR B has black wherever either A is black or B is black. Object A XOR B has black wherever either A or B but not both, is black.*

Raster-op using XOR is widely used in bitmap graphics because it is its own inverse. If a source image has been copied onto the destination using XOR, the image will be erased completely if it is copied a second time using XOR. Of course, XOR is not without its drawbacks. When a picture is drawn using XOR to position a number of objects on the plot, any overlapping portions of different objects are invisible. This leads to strange effects in scatterplots in which there is a lot of overlap. In fact, if an even number of points in a plot have exactly the same coordinates, they completely obscure each other. This makes it important to use jittering techniques in which small amounts of random noise are added to data points to avoid exact overlap (Chambers, Cleveland, Kleiner and Tukey, 1983) prior to making displays that are done with XOR mode.

To see how a dynamic display may be implemented on a bitmapped device, suppose we want to display three-dimensional data by means of rotation. Suppose, too, that the bitmapped display device has only moderate computational power and no rotation hardware. What programming techniques can we use to produce an effective display?

First, we display the points on the screen in their initial position by repeatedly using raster-op to copy an offscreen image of a plotting symbol onto the screen at plotting positions of the points. Next, we must compute the new position of each point as the point cloud turns through a small angle. This can be done by computation of a rotation matrix using trigonometric functions, composition of the rotation with the previous rotation matrix and multiplication of the data by the composite matrix. However, an approximation to these computations can be done very quickly in integer arithmetic using only shift and add instructions. This technique was used in a version of point cloud rotation by Andrews (1981) to achieve point cloud rotation on a small and slow 8-bit personal computer.

We can move each point from its present position to its new position (after rotation) by means of raster-op. For each observation we use raster-op with XOR to copy an offscreen image of the plotting symbol once again into its current position (thus deleting the point) and then use XOR again to copy the image into its new position. Thus, we delete the first point and then redraw it, delete the second point and redraw it,

and so forth through all of the points. This allows the display to move without ever visibly erasing and redrawing, although the rotating point cloud may not appear perfectly rigid when the number of points gets large.

Raster-op can also be used with extra memory to get what appears to be an instantaneous change from one rotation angle to another. Memory is allocated offscreen for a bitmap the same size as the onscreen bitmap. The points are drawn into the offscreen bitmap in their new position after rotation, and when all of the points have been drawn, the offscreen bitmap is copied into the onscreen bitmap using the raster-op with the STORE function. This may be faster than incremental plotting techniques when there are a large number of points on the display, because it is only necessary to draw each point once. Incremental techniques undraw the point and then redraw it in a new position, operating on each point twice. The drawbacks to using offscreen memory are that it requires a potentially large offscreen bitmap and it may be somewhat slower for small numbers of points because the offscreen memory must be cleared and then copied into the frame buffer.

3.4 Device-independent Graphics: Graphics Standards

It may take specialized programming techniques to get adequate performance for dynamic displays on commonly available graphics devices. Also, although some of the low level operations we described in that section are common to a number of graphics devices, actual implementations usually vary slightly from machine to machine. This means that dynamic methods whose algorithms use low level routines are usually difficult to "port" from one variety of hardware to another.

The Graphical Kernel System, known as GKS (GKS, 1985), is now an international graphics standard. There are also a number of implementations of the ACM-SIGGRAPH CORE proposal (CORE, 1977). These standards describe device-independent low level support for a variety of graphics input and output devices. They provide device independent ways of drawing lines, markers (point symbols), polygonal areas and characters. They provide a mechanism to translate from two-dimensional or three-dimensional *world coordinate systems* into coordinate systems appropriate for the graphics device. That is, they allow the programmer to specify how the natural units of measurement for the data are to be mapped onto the surface of the graphics device. The lines, markers, polygons and characters can be specialized by means of various graphical attributes such as line texture, width, color and character size, rotation and font.

Dynamic aspects of graphics are provided in the standards by a mechanism known as *segments*: a segment contains descriptions of graphical objects and also has attributes of its own such as visibility and a transformation matrix that describes how its coordinate system is to be mapped onto the screen. Graphic input is handled by a number of symbolic input types: button, valuator, locator, stroke and pick. Buttons can be pushed, valuators provide a single (continuously variable) value, locators provide single (x, y) coordinate pairs, strokes give a stream of coordinate pairs (corresponding, for example, to the positions of a mouse over time) and a pick identifies a particular segment that is visible on the display.

Data analysts should not think of graphics standards as a panacea, for there are many issues in data graphics that are not addressed by any of the proposed standards. In order to have a well defined scope, the current graphics standards have decided that they will not include applications software. This means that there is no explicit recognition of an applications area, such as data display, within the graphics standards, although the standards provide a foundation upon which good data graphics can be built. Much of the impetus for graphics standards comes from high visibility, high demand applications such as computer-aided design. Therefore, most of the effort of the computer graphics community is directed toward the realistic display of three-dimensional objects and scenes. The graphics standards contain many hooks for the display of such objects (three-dimensional perspective display of shaded surfaces, hidden surface elimination, etc). These are nontrivial problems and having good solutions in standard graphics software is quite valuable for typical applications, but the value of these features for data display applications is often much less.

Earlier we discussed how rotation could be implemented on a bitmapped display device using raster-op as the primary graphical operation. Let us try to see how graphics standards could help us to implement rotation. Using the CORE system (GKS does yet not have three-dimensional primitives), this is easy. We create a segment that uses a three-dimensional *polymarker* (points in three-space) subroutine to display our points, and then, in a loop, we change the transformation matrix to give different views of our points. That is all there is to it.

On present hardware, the results range from beautiful to disastrous. The beautiful results come on devices where the hardware provides assistance in rotating objects. On these machines, the rotation is convincing even with a large number of data points. The disappointing results come on lower cost display devices that do not have special hardware for rotation.

On these devices, the program may draw the points in one position, then erase the screen, then redraw the points in a new position. A cycle like this can take a second or two for even just a few data points. There is no illusion of a rotating three-dimensional point cloud. On the same device, the bitmapped approach can give very good results; the problem is that the operations available in the graphics standard do not map into very efficient operations on the low cost display devices. Of course, as hardware costs continue to decline, the performance of inexpensive graphics devices using standard graphics software will continue to improve. Nevertheless, it will be several years before adequate dynamic graphics is commonly available through the use of standard software.

3.5 Data Analysis Environments: Windows to the World of Data

Dynamic graphics routines, to be useful, must be embedded in a data analysis environment, which must include much more than the dynamic graphics routines. A data analyst at any time may require data management, basic graphics, statistical modeling, computations for transformations and a variety of other functions. The process of data analysis consists of jumping rapidly from one task to another as the patterns revealed in the data by one method suggest the use of another method. Thus, it is critical that dynamic methods be fully integrated with the other tools that the data analyst requires. Anything that makes it difficult to perform these rapid jumps impedes the progress of the analysis. In the past, much of the software for dynamic displays of data was implemented as standalone packages. This is appropriate for research into dynamic displays, but it is quite inadequate for use by working data analysts.

Multiwindow displays are a relatively recent development of the computer science community (Morris, Satyanarayanan, Conner, Howard, Rosenthal and Smith, 1986; Pike, 1983) that are particularly useful as part of a data analysis environment. Windows divide the surface of the display into rectangular regions, each of which operates independently. Most often the windows behave like separate terminals connected to the computer. A user can instigate a long computation in one window and then move to another window and continue to use the machine even as the long computation continues. The multiple windows allow the user to move between various tasks, just as a multitasking operating system allows the computer to work on several jobs at the same time.

The window environment is particularly well suited to data display because each graph can be in a separate window (Stuetzle, 1987). The collection of windows can be moved around on the screen, much as one would organize graphs on pieces of paper on a desk. But unlike paper graphs on a desk, the shape of windows can be changed, so a graph can be made small for temporary storage in an out-of-the-way place (or made to disappear until recalled) or a graph can have its aspect ratio changed (see Section 2.5) by changing the shape of the window. Furthermore, corresponding points on different graphs can be linked so that operations, such as brushing, performed on points in one window are reflected in other windows.

4. SUMMARY AND DISCUSSION

The amazing computing power now available to data analysts carries with it the potential for new graphical methods—dynamic graphics—that utilize visual input and achieve virtually instantaneous graphical change. High interaction methods represent a new frontier in data analysis and are an important adjunct to conventional static graphics.

Because many early dynamic graphics systems were based on rotation, they are often thought of as tools for exploring multivariate data. However, dynamic graphics methods have the potential to include all areas of data analysis. This is demonstrated by the wide range of methods discussed in Section 2.

The chief reason for the advent of dynamic graphics is the current availability of the hardware and software tools needed for their implementation. Section 3 gave an introduction to these tools, illustrating both the opportunities they afford and the limitations they impose.

One question that frequently arises about dynamic methods is how a data analyst can present results that are based on them. The methods are discovery tools that enable us to uncover interesting structures in data, but once having learned about a particular structure we can almost always design a static display to show it. This was done in this paper for several data sets. In fact, dynamic graphics can be viewed as methods that allow us to move rapidly through a long sequence of static views until the desired or interesting static view is found; having found it we can present it to others.

It is quite easy to invent a dynamic graphical method. It is very difficult to invent one that is a better tool than those already available. This paper presents basic ideas of dynamic methods and computing principles that should allow others to design new and effective dynamic methods for data analysis. The challenge is to use the known techniques as a starting point, to develop new graphical methods and to evaluate those methods as they are used on real data.

ACKNOWLEDGMENTS

Jon Bentley, John Chambers, John Hartigan, Daryl Pregibon, Werner Stuetzle and two referees gave us many useful comments on the manuscript.

REFERENCES

ANDERSON, E. (1935). The irises of the Gaspe peninsula. *Bull. Amer. Iris Soc.* **59** 2–5.

ANDREWS, D. F. (1981). Statistical applications of real-time interactive graphics. In *Interpreting Multivariate Data* (V. Barnett, ed.) 175–185. Wiley, New York.

BECKER, R. A. and CLEVELAND, W. S. (1984). Brushing a scatterplot matrix: High interaction graphical methods for data analysis. AT&T Bell Laboratories Technical Memorandum, Murray Hill, N. J.

BECKER, R. A. and CLEVELAND, W. S. (1987). Brushing scatterplots. *Technometrics* **29** 127–142.

BECKER, R. A., CLEVELAND, W. S. and WEIL, G. (1987). Some comments on brushing and rotation. In *Dynamic Graphics for Statistics* (W. S. Cleveland and M. E. McGill, eds.). Wadsworth, Pacific Grove, Calif. In press.

BECKER, R. A. and MCGILL, R. (1985). Dynamic displays of data (video tape). AT&T Bell Laboratories Technical Memorandum, Murray Hill, N. J.

BRUNTZ, S. M., CLEVELAND, W. S., KLEINER, B. and WARNER, J. L. (1974). The dependence of ambient ozone on solar radiation, wind, temperature, and mixing height. In *Symp. on Atmospheric Diffusion and Air Pollution* 125–128. Amer. Meteorological Soc., Boston.

BUJA, A., ASIMOV, D., HURLEY, C. and MCDONALD, J. A. (1987). Elements of a viewing pipeline for data analysis. In *Dynamic Graphics for Statistics* (W. S. Cleveland and M. E. McGill, eds.). Wadsworth, Pacific Grove, Calif. In press.

CHAMBERS, J. M., CLEVELAND, W. S., KLEINER, B. and TUKEY, P. A. (1983). *Graphical Methods for Data Analysis.* Wadsworth, Belmont, Calif.

CLEVELAND, W. S. (1985). *The Elements of Graphing Data.* Wadsworth, Monterey, Calif.

CLEVELAND, W. S. and DEVLIN, S. J. (1986). Locally-weighted multiple regression: An approach to regression analysis by local fitting. *J. Amer. Statist. Assoc.* To appear.

CLEVELAND, W. S. and MCGILL, R. (1987). Graphical perception: The visual decoding of quantitative information on displays of data (with discussion). *J. Roy. Statist. Soc. Ser. A* **150** 192–229.

CORE (1977). Status report of the graphics standards planning committee of ACM/SIGGRAPH. *Comput. Graphics* **11**.

CRILE, G. and QUIRING, D. P. (1940). A record of the body weight and certain organ and gland weights of 3690 animals. *Ohio J. Sci.* **15** 219–259.

DAVIES, O. L., BOX, G. E. P., COUSINS, W. R., HINSWORTH, F. R., HENNY, H., MILBOURN, M., SPENDLEY, W. and STEVENS, W. L. (1957). *Statistical Methods in Research and Production.* Oliver and Boyd, London.

DIACONIS, P. and FRIEDMAN, J. H. (1980). M and N plots. Technical Report 15, Dept. Statistics, Stanford Univ.

DONOHO, A. W., DONOHO, D. L. and GASKO, M. (1985). *MACSPIN Graphical Data Analysis Software.* D^2 Software, Austin, Texas.

DONOHO, D. L., HUBER, P. J., RAMOS, E. and THOMA, H. M. (1986). The man-machine-graphics interface for statistical data analysis. In *Statistical Image Processing and Graphics* (E. J. Wegman and D. J. DePriest, eds.). Dekker, New York.

FISHER, R. A. (1936). The use of multiple measurements in taxonomic problems. *Ann. Eugenics* **7** 179–188.

FISHERKELLER, M. A., FRIEDMAN, J. H. and TUKEY, J. W. (1975). PRIM9: An interactive multidimensional data display and analysis system. In *Data: Its Use, Organization, and Management.* 140–145. Association for Computing Machinery, New York.

FOWLKES, E. B. (1971). User's manual for an on-line interactive system for probability plotting on the DDP-224 computer. AT&T Bell Laboratories Technical Memorandum, Murray Hill, N. J.

FRIEDMAN, J. H., MCDONALD, J. A. and STUETZLE, W. (1982). An introduction to real time graphical techniques for analyzing multivariate data. *Proc. Third Ann. Conf. and Exhibition of the National Computer Graphics Assoc.* 421–427.

GKS (1985). *Graphical Kernel System (GKS): ANSI X3.124-1985.* American National Standards Institute, New York.

GOULD, S. J. (1979). *Ever Since Darwin: Reflections in Natural History.* Norton, New York.

HECKBERT, P. S. (1984). Techniques for real-time frame buffer animation. *Comput. FX '84* 57–67.

HUBER, P. J. (1983). Statistical graphics: History and overview. *Proc. Fourth Ann. Conf. and Exposition of the National Computer Graphics Assoc.* 667–676.

JERISON, H. J. (1955). Brain to body ratios and the evolution of intelligence. *Science* **121** 447–449.

KEELING, C.D., BACASTOW, R. B. and WHORT, T. P. (1982). Measurement of the concentration of carbon dioxide at Mauna Loa Observatory, Hawaii. In *Carbon Dioxide Review: 1982* (W. C. Clark, ed.) 377–385. Oxford Univ. Press, New York.

LITTLEFIELD, R. J. (1984). Basic geometric algorithms for graphic input. *Proc. Fifth Ann. Conf. and Exposition of the National Computer Graphics Assoc.* 767–776.

MCDONALD, J. A. (1982). Interactive graphics for data analysis. Technical Report, Dept. Statistics, Stanford Univ.

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

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

MORRIS, J. H., SATYANARAYANAN, M., CONNER, M. H., HOWARD, J. H., ROSENTHAL, D. S. H. and SMITH, F. D. (1986). Andrew: A distributed personal computing environment. *Comm. ACM* **29** 184–201.

NEWMAN, W. M. and SPROULL, R. F. (1979). *Principles of Interactive Computer Graphics.* McGraw-Hill, New York.

NEWTON, C. M. (1978). Graphics: From alpha to omega in data analysis. In *Graphical Representation of Multivariate Data* (P. C. C. Wang, ed.) 59–92. Academic, New York.

OTTENWELLER, J. E., TAPP, W. N. and NATELSON, B. H. (1986). Biological aging in the cardiovascular system of syrian hamsters. Dept. Neurosciences Technical Report, VA Medical Center, N. J. Medical School, Newark, N. J.

PIKE, R. (1983). Graphics in overlapping bitmap layers. *ACM Trans. Graphics* **2** 135–160.

PIKE, R., GUIBAS, L. and INGALLS, D. (1984). Bitmap graphics: SIGGRAPH'84 course notes. AT&T Bell Laboratories Technical Memorandum, Murray Hill, N. J.

STUETZLE, W. (1987). Plot windows. *J. Amer. Statist. Assoc.* **82** 466–475.

TUFTE, E. (1983). *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, Conn.

TUKEY, J. W. (1973). Some thoughts on alternagraphic displays. Technical Report 45, Series 2, Dept. Statistics, Princeton Univ.

TUKEY, J. W. (1982). Another look at the future. *Comput. Sci. Statist. Proc. 14th Symp. Interface* 2–8.

TUKEY, J. W. (1987a). Control and stash philosophy for two-handed, flexible, and immediate control of graphic display. In

The Collected Works of John W. Tukey (W. S. Cleveland, ed.) **5**. Wadsworth, Pacific Grove, Calif.

TUKEY, J. W. and TUKEY, P. A. (1985). Computer graphics and exploratory data analysis: An introduction. *Proc. Sixth Ann. Conf. and Exposition of the National Computer Graphics Assoc.* 773–785.

TUKEY, J. W. and WILK, M. B. (1965). Data analysis and statistics: Techniques and approaches. In *Proc. Symp. Information Processing in Sight Sensory Systems* (P. W. Nye, ed.) 7–27. California Institute of Technology, Pasadena, Calif.

YULE, G. U. (1927). On the method of investigating periodicity in disturbed series, with special reference to Wolfer's sunspot numbers. *Philos. Trans. Roy. Soc. London Ser. A* **226** 267–298.

Comment

John W. Tukey

The overall impact of this paper is both substantial and sound. Thus my comments will have to focus on recommended changes in flavoring or on possibilities for the future.

1. THE MUD CAN BE DULL

The Murray Hill tradition in data analysis has long included aspects of "plant your feet firmly on the ground, even if they do sink deep in the mud." The limitation of scatterplot matrices to original coordinates is a case in point. The discussion of (brain weight)/(body weight)$^{2/3}$ in Section 2.1 is another case in point. A scatter of "log brain weight MINUS ⅔ log body weight" against log body weight would be a useful supplement to the scatter of Figure 2, in part because it would offer an expanded vertical scale. In Figure 11, where "abrasion is stated to be the intended response," an additional row and an additional column for "abrasion loss residual" and "tensile strength" would greatly clear up the situation—perhaps leaving brushing the task of finding still subtler behavior.

2. HIGHLIGHTING MAY BE INESCAPABLE—BUT IS STILL INADEQUATE

Paper representations of screens with highlighted points are rather weak and wan—and highlighted screens may be somewhat so. Particularly for paper versions, we ought to further enhance the contrast between emphasized and background points. Two easy ways to do this are: a) median +'s or x's for emphasis, with dimensions at least 3 times those of background circles, or b) filled circles for emphasis and little dots for background. This sort of improvement is needed for alternagraphic emphasis as well as for brushed emphasis. (Compare with the last paragraph of Section 2.1 in Becker, Cleveland and Wilks.)

3. DO PANELS MAKE UP A TABLE OR A GRAPH?

To Becker, Cleveland and Wilks, the answer seems to be "clearly, a table!" because they number rows of panels from above down (and put the vertical coordinate first!). For some of us, the answer seems to be "clearly, a graph" so we would number rows of panels from below upward, and put the horizontal coordinate first.

Whichever side you take—if one is to write in text about panel numbers, panel rows and panel columns, in the pictures, they should be labeled clearly enough (e.g. (1, −) and (−, 3) *or* (−, 1) and (3, −)) so it would be really hard for the reader/viewer to miss the point.

If you do adhere to the graphic paradigm, the distinctive diagonal of your scatter-plot matrix will run NE-SW and not NW-SE.

4. RECTANGLES, ANYONE?

It would have been helpful if the account of brushing had said—if it is true, as I think—that brushes are rectangular because rectangles can be computed faster. How much faster? Are we near the present boundaries when we include brushing? Or could we afford other brush shapes?

5. COGNOSTICS, ANYONE?

The paragraph in Section 2.4 on the scatterplot matrix assumes that scrolling is our only remedy when p is too large. Another approach would be to use cognostics (e.g., Tukey and Tukey, 1985) to help us rearrange our variables so that the initial view shows the most interesting k of them. Instead of simple scrolling, then, we might hold the first $k - 2$ (or $k - 3$, or $k - 1$) fixed and scroll the other 2 (or 3 or 1).

John W. Tukey is Senior Research Statistician and Donner Professor of Science, Emeritus, at Princeton University. His mailing address is 408 Fine Hall, Washington Road, Princeton, New Jersey 08544.