Guiding Data Analysis with Visual Statistical Strategies[1]

Forrest W. Young
Psychometric Laboratory
University of North Carolina
Chapel Hill, NC, USA

David J. Lubinsky
Department of Computer Science
University of Witwatersrand
Johannesburg, South Africa

Abstract

The concept of statistical strategy is introduced and used to develop a structured graphical user interface for guiding data analysis. The interface visually represents statistical strategies that are designed by expert data analysts to guide novices. The representation is an abstraction of the expert's concepts of the essence of a data analysis. We argue that an environment that visually guides and structures data analysis will improve data analysis productivity, accuracy, accessibility and satisfaction in comparison to an environment without such aids, especially for novice data analysts. Our concepts are based on notions from Cognitive Science, and can be empirically evaluated.

The interface consists of two interacting windows: the *guidemap* and the *workmap*. Each window contains a graph which has nodes and edges. The guidemap graph represents the statistical strategy for a specific statistical task (such as describing data). Nodes represent potential data-analysis actions that can be taken by the system. Edges represent potential actions that can be taken by the analyst. The guidemap graph exists prior to the data-analysis session, having been created by an expert. The workmap graph represents the complete history of all steps taken by the data analyst. It is constructed during the data-analysis session as a result of the analyst's actions. Workmap nodes represent datasets, data models, or data-analysis procedures which have been created or used by the analyst. Workmap edges represent the chronological sequence of the analyst's actions. One workmap node is highlighted to indicate which statistical object is the focus of the strategy. We illustrate our concepts with ViSta, the Visual Statistics system that we have developed.

Keywords: Artificial Intelligence, Cognitive Science, Hypertext, Lisp-Stat, Program Visualization, Statistical System Design, ViSta, Visual Programing

1. The original concepts discussed in this paper were conceived jointly by both authors. Michael Frigge made a first attempt at an implementation, which was later abandoned. The implementation and development described in this paper were done entirely by the first author, who was also the primary author of the paper. The final version of this paper has benefitted from many constructive discussions with Richard A. Faldowski. Contact the first author for information or for software at forrest@unc.edu, or at UNC Psychometric Laboratory, CB-3270 Davie Hall, Chapel Hill NC 27599-3270, USA.

1.0 Motivation

Data are the lifeblood of science. Because computerized data-analysis systems help scientists understand data, they have become of central importance to the scientific enterprise, evolving into extensive and powerful systems capable of performing many kinds of very sophisticated and complex analyses.

Unfortunately, the structure of data-analysis systems has evolved willy-nilly over the years. While much thought has been focused on the kinds of analyses that can be performed by these systems, less thought has been given to their overall structure: It seems to us that the more powerful a statistical system is, the more clumsy it is to use.

At the same time that these changes have been taking place in data-analysis software, the computational hardware on which these systems run has increased in capability: Processors are faster, memory is larger, displays are bit-mapped instead of character-mapped, and mice are widely use. And all the while, cost has decreased.

Thus, compared to a few years ago, much more complex statistical analyses can now be performed in less time and at less cost. Furthermore, a much broader cross-section of the scientific community has come to have access to data-analysis systems. However, the main difficulty with essentially all data-analysis systems is not the sophistication of the analyses that can be performed. Rather, it is the fact that these systems are not designed with the capabilities of the user in mind. This weakness makes the systems difficult to learn and use, especially for the new community of novice users.

While much effort and thought have been expended to improve the capability of individual components of many statistical systems, relatively little effort and thought have been given to providing data-analysis systems with an overall structure that would make them accessible to the full range of users, from novices to experts. Some systems, especially the older systems such as SAS (SAS Institute, 1990) or SPSS (Norusis, 1990) seem to have no overall unifying design and little regard for the capabilities of their users. Other systems, such as S (Becker, Chambers & Wilks, 1988), are designed with the capabilities of the sophisticated user in mind, being, essentially, high-level data-analysis languages. Recently available systems such as DataDesk (Velleman & Velleman, 1988) and JMP (SAS Institute, 1990) are designed from the ground up on the basis of a unified graphical user interface metaphor. As one might expect, these systems are indeed much more appropriate for the novice user. However, these systems have no data-analysis language, and so lack the flexibility and customizability that the sophisticated user needs. The Lisp-Stat (Tierney, 1990) and APL2STAT (Friendly and Fox, 1994) systems seem to have the potential to be appropriate vehicles for implementing interfaces for the novice and the expert, since they are designed with an object-oriented and language-based philosophy, and since they are extensible. However, both lack a graphical user interface.

In all statistical systems that we are familiar with, even when simple data-analysis procedures are used, novice users are soon at a loss as to how to combine several data-analysis procedures into a cogent statistical strategy that reveals the basic information in the data. The very power of many systems can actually hinder the data-analysis task, especially for users who are novices. It appears to us that there is a paradox: For many users increasingly powerful and sophisticated data-analysis systems have become less suited for understanding data than their less powerful and simpler forebears.

In this paper we propose that data-analysis environments should present visualized statistical strategies and structures. We present an environment that guides the data-analysis steps taken by novice data analysts. Our environment also aids data analysts at all levels of sophistication by showing them the structure of their analysis session. In addition, sophisticated users can perform analyses simply by using the underlying data analysis language, if they don't want to use the graphical interface. Finally, our environment includes graphical tools that can be used by expert data analysts to create the analysis strategies that are used to guide novice analysts.

Our fundamental hypothesis is that data analyses performed by data analysts in an environment that visually guides and structures the analysis will be more productive, accurate, accessible and satisfying than data analysis performed by the same people in an environment without such visual aids. We hypothesize that this should be true for all data analysts, but more so for novices. We also hypothesis that novices will choose to use the aids more often than those with more experience. We realize that these are hypotheses, and that we have no evidence to support them. While we do not test them in this paper, we do propose methods for doing so.

The implementation of our ideas is called ViSta, a visual statistics system developed by the first author (Young, 1994). ViSta, which is written in Lisp, adds our structured graphical user interface to Lisp-Stat's (Tierney, 1990) statistical engine, object system, graphical system and windowing environment. Lisp-Stat has been extensively reviewed by a number of authors (Baxter & Cameron, 1991; Lubinsky, 1991; Weihs, 1991; Young, 1991), and is the basis of at least one other extensive development project (Cook & Weisberg, 1994). ViSta is available from the first author, or can be downloaded from ftp.stat.ucla.edu. A more complete version of this paper is available (Young & Lubinsky, 1994). We are very interested in feedback about the efficacy of our ideas and implementation.

2.0 Background

We hold that data analysis is a highly complex activity (Young & Smith, 1991) that involves repetitive actions that occur over and over again: Data analysis is a repetitive, cyclical search for understanding (Lubinsky & Pregibon, 1988). We believe that data analysis productivity, accuracy, accessibility and satisfaction will improve in an environment that guides and structures the actions that occur during the search for meaning in data.

One of our main design principles is that a data-analysis system should incorporate a variety of environments, each suited to a specific level of data analysis sophistication that a user might have, so as to maximize the data analyst's productivity and satisfaction. We believe that data-analysis software should be designed to accommodate the complete range of data analyst sophistication, from novice to expert.

We identify four kinds of data analysts: novice, competent, sophisticated and expert. Accordingly, we propose four kinds of environments: First, there should be *guidemaps* to guide novice data analysts through complete data analyses; second, there should be *workmaps* to inform novice and competent data analysts of the overall structure of their data-analysis sessions; third, there should be an underlying *data analysis language* to let sophisticated data analysts dispense with the visual aids when they find them unnecessary; finally, there should be an *authoring mode* to help expert data analysts create the guidance diagrams that are used by novices. In addition to these four environments, which are all highly interactive, there should be a script environment for automating repetitive data analyses. These five environments should be seamlessly integrated within the statistical analysis environment. Analysts should be able to easily switch between them when desired, as we believe that analysts do not have the same level of expertise for all aspects of data analysis.

Structuring Data Analysis: Young & Smith (1991) argue that the process of data analysis is improved when the environment structures the actions taken by the data analyst. They suggest that an evolving data analysis should be represented by an icon-based graphical user interface which constructs a map of the analysis as it proceeds. This map shows the structure of the actions taken by the data analyst, and the data, models and analysis procedures involved in those actions. The map presents the analyst with a visualization of the structure of the analysis session, and can be used to return to previous steps.

For our work, the formal representation of session structure is the workmap:

Definition: A *workmap* is a directed acyclic graph consisting of nodes and edges (as suggested by Young & Smith, 1991), where a node represents a data-analysis object (a dataset or a data model) or a data-analysis procedure that has been used by the analyst, and an edge represents the chronology sequence of the objects and procedure (the creation dependencies) during the analysis session. Taken as a whole, the workmap is a visual, object-oriented, directly manipulable, structured representation of the history of a data-analysis session.

Notice that a node is a self-contained unit of existing data (dataset), statistical computation (analysis procedure), or a combination of the two (data model), whereas edges represent the choices, actions and decisions that a data analyst made during the session. Nodes, which are the basic building blocks of the evolving data-analysis session, can be selected and reviewed at any time. The workmap visualizes the history of the evolving data analysis. It is a realization of a specific statistical strategy. (Workmaps have been incorporated in the BBN/Cornerstone system (BBN Software, 1994) in consultation with the first author).

Guiding Data Analysis: Data analysis is a complex activity that involves many steps, and at each step the data analyst is faced with many choices. Often, the data analyst returns to previous steps in order to make different choices. As stated by Lubinsky and Pregibon (1988), “Like a detective, a data analyst will experience many dead ends, retrace his steps, and explore many alternatives before settling on a single description of the evidence in front of him.” We argue that data analysis will improve when it occurs in an environment that guides the actions taken by the analyst to understand data.

We use the Artificial Intelligence notion of strategy as a basis for developing methods for guiding data analysts. Several statisticians, notably Chambers (1981), Gale and Pregibon (1982), Gale (1988), Oldford and Peters (1988), Pregibon and Gale (1984), Hand (1984; 1985), Lubinsky and Pregibon (1988), Van den Berg (1992), Gale, Hand & Kelley (1993), Hand (1993) and Hand (1994), have worked on the notion of a statistical strategy. We use this body of work, which is extensively reviewed by Gale, Hand & Kelly (1993), to develop our own definition of statistical strategy:

> **Definition:** A *statistical strategy* is a formal representation of an expert statistician's conceptual structuring of 1) the data-analysis *procedures* to accomplish a specified data-analysis task; 2) the data analyst's *actions* (choices, decisions, etc.) that are possible with the procedures; and 3) the relationships between the procedures and actions needed to accomplish the task. The data-analysis task is to understand a specified data-analysis object (a dataset or data model).

For our data-analysis environment guidemaps are the formal representation of statistical strategy:

> **Definition:** A *guidemap* is a directed cyclic graph consisting of nodes and edges. The nodes of the graph represent data-analysis procedures, whereas the edges represent the analyst's possible actions. The structure of the map indicates the order dependencies between the procedures and the actions that can be taken with the procedures to accomplish the data-analysis task of understanding the data-analysis object. Finally, the data-analysis object (dataset or data model) is represented by a highlighted node of the *workmap*. It is said to be the focus object.

Notice that a guidemap node is a self-contained unit of *potential* statistical computation, while a guidemap edge represents the expert's guidance about moving from one computation to the next. Nodes are the basic building blocks of potential data analyses, i.e., of statistical strategies. On the other hand, the edges in the strategy represent the data analysts's possible choices, actions and decisions regarding the use of data-analysis procedures. They indicate permissible paths for traversing the nodes. Nodes can only be selected when they are highlighted. As a whole, the guidemap visualizes and abstracts the essence of an expert's statistical strategy.

3.0 Representing Statistical Strategy

In this section we discuss our definition of statistical strategy in detail, focusing on the four key aspects of the definition: the formal representation; the data-analysis object that is the focus of the strategy; the role of the expert statistician; and objects, procedures and actions.

3.1 The Formal Representation of Strategy

First, our definition states that a statistical strategy is based on a *formal representation*. Our formal representation consists of graph structures like that shown in the guidemap window of Figure 1 (a screen image from ViSta). The

Insert Figure 1 About Here

guidemap, titled **Analysis Cycle**, presents the overall statistical strategy. This specific guidemap is always the first guidemap for a newly created dataset object. It is only a small portion of the overall strategy, since it causes additional “sub”-guidemaps to be displayed in the window. Taken as a whole, the guidemap in Figure 1, plus all of the additional guidemaps, are our formal representation of statistical strategy.

The strategy concerns a specific data or model object, thus, a data or model object is the focus of the analysis. The focus object is represented in the workmap window by the highlighted (dark) icon. The workmap itself shows where this object fits into the structure of the overall evolving analysis. The two separate windows emphasize the separation between the evolving data analysis (mapped in the workmap) and the strategy that is guiding the data analysis (mapped in the guidemap). We discuss the workmap in the next subsection. Here, we discuss the guidemap.

As stated above, the guidemap is a directed (possibly) cyclic graph consisting of edges and nodes. In our work, guidemap nodes are represented by the rectangular *button* icons, and guidemap edges are represented by the *arrows*. Thus, the buttons show potential steps in the analysis that the analyst is guided to take, whereas the arrows indicate the flow of guidance from one step to the next. A node is a self-contained unit of *potential* statistical computation which may do its own computations, or, recursively, call another strategy.

Buttons can be “active” or “inactive”. Active buttons are highlighted (such as the **Link:Explore** button in Figure 1) and are ready to cause an action. Clicking on the **??** side of an active button enters a hypertext which causes help to be displayed about the action of the button. Clicking on the **!!** side of an active button enters a hypercode which causes the button’s action to be initiated. Once the button’s action has taken place, the highlighting (activation) of the buttons changes: The clicked button deactivates, and the buttons that it points to are activated. Inactive buttons (such as the **Link:Transform** button in Figure 1) are not ready to do anything: Clicking on them has no effect.

There are two kinds of buttons: Flow Buttons, which control the flow between various portions of the large structure of guidemaps, and Procedure Buttons, which control the use of data-analysis procedures.

Flow buttons include the **Link**, and **GoTo:Model** buttons in Figure 1, and the **Return** and **GoTo:Data** buttons in Figure 2. These buttons take the user to other guidemaps. The **Link** button takes the analyst to a new strategy, whereas

Insert Figure 2 About Here

the **Return** button returns to the linked-from strategy. The **Link** button is, in essence, a *macro* data-analysis procedure which is itself a strategy, since this button opens up new strategies. For example, clicking on the **!!** portion of the **Link:Explore** button in Figure 1 causes the **Explore Data** guidemap, shown in Figure 2 to appear. Correspondingly, clicking on the **Return** button in Figure 2 will take you back to Figure 1. Upon return to the guidemap in Figure 1, the highlighting of the buttons will change according to the connecting arrows. That is, the **Link:Explore** button will deactivate, and the **Link:Transform** and **Link:Analyze** buttons will activate.

The **GoTo** button changes the focus of the data analysis, and of the strategy, to a new data or model object. When a new object has been created and named, then the name of that object replaces **Data** or **Model** in the **GoTo** button. Then, when the **GoTo** button is clicked, the appropriate data or model icon is highlighted in the workmap, and the appropriate strategy is displayed in the guidemap window.

All buttons other than flow buttons are procedure buttons that activate data-analysis procedures. In Figure 2 we see procedure buttons such as **List Variables** and **Visualize Data**. When an active procedure button is clicked, the indicated data-analysis procedure (listing variables, showing the datasheet) is activated.

3.2 The Focus of the Strategy

The focus of a statistical strategy is a data-analysis object (a dataset or a model). In Figure 1, the icons named **CarRatings** and **Norm-CarRatings** are data icons, whereas **PCA-CarPrefs** is a model. The focus object is represented by the icon that is highlighted in the workmap.

Each time a new object is created, it is represented by a new icon. Whenever a new dataset or model object is derived from an existing dataset object, an arrow is drawn from each of the new object's parents (usually only one) to the new object to show the creation dependency. These arrows have a meaning that parallels, but is somewhat different from, their meaning in the guidemap: They represent the flow of data into or out of a data-analysis object (dataset or model) or procedure as a result of a data analyst's action. In the guidemap, on the other hand, a arrows represent potential actions a data analyst might take.

The evolving progress of the data-analysis session is shown in the workmap. Certain actions taken via the guidemap create new nodes in the workmap. A new dataset object may be created by a mathematical procedure (such as normalization or principal components analysis) or by a non-mathematical operation (such as removing variables or merging datasets). A new model object is always created by a mathematical procedure. A procedure icon appears between the original and new objects when the creation involved mathematical operations, otherwise, no procedure icon appears. If a procedure icon appears, the creation dependency arrow is drawn from the parent objects through the procedure to the new object. Naturally, a new object may be brought in from "outside" of the system, in which case the new object is not connected to a parent (e.g., **CarRatings** in Figure 1).

The specific object which is the focus of the analysis (and, therefore, of the analytic strategy) is highlighted in the workmap. In Figure 1, **Scores & Ratings** is the focus object. Any data or model object in the workmap can be selected at any time to be the new focus object. When a new focus object is selected, the new strategy associated with it is displayed in the guidemap window, and the user enters that strategy.

The workmap and guidemap graphs differ in several respects. First, the structure of the guidemap graph doesn't change, it remains as shown throughout the analysis, although its highlighting changes. The workmap graph, on the other hand, grows as new data and model objects are created and as new analysis procedures are used (both structure and highlighting change). Second, the guidemap is a (potentially) cyclic graph, whereas the workmap is an acyclic hierarchical tree graph. This represents Lubinsky and Pregibon's (1988) observation that actions taken during data analysis are not hierarchical, but are cyclical, although the resulting analysis is hierarchical. Third, the guidemap (as represented by the initial guidemap shown in Figure 1, and all its sub-guidemaps) has an entry point but no exit point, whereas workmaps have both entries and exits. This represents the fact that a strategy has a beginning step but no final step. The lack of an exit point from a strategy reflects the fact that a strategy is cyclic, and that users should be able to quit a strategy (with the window's close box) whenever they choose.

3.3 The Role of the Expert Statistician

We turn now from the first two aspects of our definition of strategy (the formal representation and its focus) to the third aspect, namely that a statistical strategy represents the conceptual structure of an expert statistician.

It is assumed that the expert is only expert in a proscribed domain of statistical analysis, not for the entire domain. The role of such an expert is to decide, for the expert's area of statistical analysis expertise, what steps are involved, and in what order the steps should be taken. Thus, the representation shown in the guidemap in Figures 1 and 2 (and in other guidemaps that are not shown) is an experts knowledge about exploratory data analysis. These guidemaps represent the expert's conceptual structure of the sequence of steps involved in exploratory data analysis. The expert creates these guidemaps by using the "guidetools" that are discussed in Section 5.0.

3.4 The Objects, Procedures and Actions

The final aspect of our definition of strategy is that the expert's conceptual data analysis structure concerns three classes of things and the relationships among these things. The things are the *data-analysis objects*, the *data-analysis procedures*, and the *data analyst's actions*. All three are included in our representation of statistical strategy.

Data-analysis Objects: There are two types of data-analysis objects: dataset objects and model objects. Both types of data-analysis objects are represented by icons in the workmap (but not in the guidemap). Datasets are represented by tall rectangular icons containing very narrow vertical bars (representing variables). Models, like data, are represented by tall rectangular icons, but they contain mathematical symbols as well as "variable" bars to reflect the fact that models are data that have been subjected to mathematical operations. The highlighted data-analysis object is the focus of the statistical strategy.

Data-analysis Procedures: Procedures are represented by the wide rectangular icons in the workmap and guidemap windows[1]. The procedures are the nodes of the guidemap's strategy structure, with each node being a self-contained piece of statistical computation, including visualizations (construction and presentation of dynamic statistical graphics), tables and textual results. These procedure-nodes, in the exploratory data analysis example shown in Figure 2, include a datasheet, the ability to list variables and observations, data visualization and reporting, summary statistics, and the ability to create new data. These are the kinds of exploration procedures the expert deems to be appropriate parts of the analysis strategy.

Data Analyst's Actions: The possible actions of the data analyst are represented in the guidemap by the arrows connecting the procedure icons. On the other hand, in the workmap the arrows indicate actions that the data analyst has already taken. In the guidemap window, the direction of the arrow indicates the order in which the expert thinks the novice should use the data-analysis procedures. Thus, the data exploration strategy in Figure 2 indicates that the expert thinks the first three steps should be looking at the data themselves or listing their variable names or observation labels. Note that these procedure-buttons are highlighted and others are not. Once all three of these actions are taken, the next three buttons become highlighted (and the first three become gray), indicating that the next three analysis procedures are now available. In this way, the novice is guided through the data exploration strategy. At least one of the procedure-buttons in the guidemap window is always active, indicating which of the procedures can be used next by the analyst. Initially, when a strategy is entered, certain procedure(s) are highlighted, indicating what the analyst should do, and that the system is waiting for an action.

4.0 Using Statistical Strategies

In the previous section we described our visual representation of statistical strategy, a representation involving two graphs called the guidemap and workmap. In this section we describe how the data analyst uses these two graphs.

4.1 Using the Guidemap

The guidemap window presents a map of an expert's statistical strategy. This map is used to guide data analyses performed by novice analysts. At the very beginning of the analysis of a new dataset object (see Figure 1) the guidemap window contains the **Analysis Cycle** guidemap. This guidemap presents the overall flow of a data analysis, emphasiz-

1. Only the major data-analysis procedures, such as the "PrnCmp" principal components procedure, are represented in the workmap by icons (see Figure 1). This avoids cluttering the workmap. However, all procedures are iconized in the guidemap, to maximize detail. For this reason, only small portions of the entire guidemap are presented to the data analyst at any one time.

ing the major steps and their cyclical relationship. The initial highlighting of this map guides the user to explore the data, since the **Link:Explore** button is the only active (highlighted) button.

The flow of guidance is indicated by the arrows connecting buttons: When an active button's action is completed, the button deactivates (changes to gray), and the buttons that are pointed to by its arrows are activated. The change in highlighting indicates the actions that the user is guided to take next, and the arrows indicate how guidance flows. Therefore, in Figure 1, after the data are explored the analyst is guided to transformation or analysis.

Let's consider how the guidemap in Figure 1 works. First of all, note that all of the buttons in the guidemap are *macro* buttons: Whenever one of them is used a new strategy map will replace the one shown in the figure. When the new strategy map is completed, the user will once again be shown the map in the figure, although it's pattern of highlighting will have changed as indicated by the arrows. Thus, after exploring the data, the transformation and analysis (i.e., model fitting) buttons become highlighted. If transformation is chosen first, then when this is completed the analyst will be guided to analyze the data. If, instead, analysis is chosen before transformation, then when the analysis is complete the **GoTo:Model** button will become highlighted. Note, however, that if the **Transform** button was not used before the analysis, it will remain highlighted, so that the user now has the choice of either transforming the data and then reanalyzing, or of proceeding to look at the model. Finally, after looking at the model, the user can either transform the data once again, or start over with a new set of data. Thus, this map represents the expert's view that data analysis is a cycle that begins with exploration and which may or may not involve transformation before the first data analysis (model fitting). Then, the model resulting from the analysis should be looked at. The model may or may not suggest re-transformation, with this cycle of transformation, analysis and model inspection continuing indefinitely.

Note that when a new dataset object is created (for example, by transformation) the user will always be given the choice to change the focus of the strategy to the new data, thus beginning the analysis cycle all over again with a brand new, unused **Analysis Cycle** guidemap, starting with data exploration. On the other hand, the analyst may also continue focusing on the old data, if desired, although usually when new data are created the user will shift focus to them. Thus, there is an implicit cycle in the data analysis process that does not appear in the guidemap: Whenever new data are created the analysis cycle usually recommences. Actually, it might be better to describe data analysis as a *spiral*, rather than a cycle, since when the analysis cycles back around it is usually at a deeper level of analysis with a new focal dataset.

Let us now turn to consider what happens when the **Link:Explore** button is used. Since this button is a *macro* button (i.e., a button which corresponds to another guidemap), when it is used the map in the window changes to the **Explore Data** guidemap shown in Figure 2. Now, as indicated by the button highlighting, the analyst has the choice of three actions: show the datasheet, list variable names or list observation labels. When the user chooses any one of these three actions, the action takes place and the chosen button turns gray, since it is no longer a recommended action. The other two buttons remain highlighted.

Notice that the just-used button is connected to a short vertical arrow rather than to another button. This short vertical arrow is called an *and* icon because it is an “and gate” that restricts the flow of guidance from one action to the next. Specifically, all of the buttons that are connected *to* an *and* icon must be used before guidance can flow through the icon to the buttons that follow it.

Thus, when one of the active buttons in Figure 2 is used, no other buttons become highlighted until all three active buttons are used. Then, all of the buttons that have arrows pointing to them from the *and* icon are activated. In this way the user is guided to use all three active buttons in Figure 2 before doing anything else. They can be used in any order. Once they are all used, the next group of three buttons is activated, and the analyst must use them (in any order) before going on. After these three buttons have been used, the guidemap appears as shown in Figure 3.

Insert Figure 3 About Here

The guidemap in Figure 3 has changed from the one in Figure 2: The data analyst is now being guided to either return to the guidemap which led to this one (the one shown in Figure 1, but with the **Transform** and **Analyze** buttons activated) or to create a new dataset object. The analyst may wish to take the latter step to create a subset of the original data. If the decision is made to create new data, then the analyst has the choice of going to those data, which brings up a brand new **Analysis Cycle** map (identical to that shown in Figure 1) or of returning to the old **Analysis Cycle** map (with the structure shown in Figure 1, but with **Transform** and **Analyze** activated).

Note how the strategy has guided the analyst: As shown in Figure 1, the analyst must explore the data first. The analyst must analyze the data before inspecting the model. In Figure 2 and 3 the analyst must look at the data and their identifying information before visualizing the data or getting summary statistics. On the other hand, the data analyst has choices: In Figure 1, it is not required, though it is possible, to transform the data before fitting the model. Similarly, in Figure 2, it is possible to visualize the data before seeing summary statistics, or to do the actions in the reverse order.

4.2 Using the WorkMap

In the example shown in Figure 1, the workmap shows a data analysis session that has already involved several major steps. In the first step, the analyst read in the data that defined the **CarPrefs** dataset object. These data were then submitted to a Principal Components Analysis, as indicated by the **PrnCmp** procedure icon. This analysis produced the **PCA-CarPrefs** model object. The analyst then requested that a new dataset object **Scores-PCA-CarPrefs** be created by the model object. Separately, the analyst also read in data that defined the **CarRatings** dataset object. These data were normalized, as indicated by the **Norm** procedure icon, creating a new dataset object named **Norm-CarRatings**, which was merged with the **Scores-PCA-CarPrefs** dataset object to obtain another dataset object named **Scores & Ratings (**the current focus of the statistical strategy).

It should be emphasized that portions (or all) of the data analysis can be created directly in the workmap window, without using the guidemap window, whenever a sufficiently sophisticated data analyst wishes. An entire data analyses can be created from the workmap without ever seeing a guidemap. This can be done by clicking the mouse on the body of an icon to obtain a popup menu of actions that the icon supports.These menu-items are also accessible from the menubar shown.

It should also be emphasized that a previous portion of the data analysis can be revisited at any time by simply clicking on the appropriate workmap icon. Then, the analysis can be continued in a new direction by simply taking different steps than were taken previously. The workmap graph provides a very convenient and simple way of backtracking, a feature that can be very hard to do with conventional systems which do not keep the full history of a data analysis session. Note that this can be done across sessions by simply saving the workmap (or portions of it) and reloading it during another session.

Also, note that if the data analyst is performing the analysis directly from the workmap, that guidance is available at anytime by simply requesting that the guidemap be shown. When so requested, the appropriate portion of the guidemap structure is displayed in the guidemap window. Thus, it is possible for the data analyst to use guidance when needed, and to avoid it when it is not needed.

4.3 Data Analysis without Guidemaps and Workmaps

Finally, portions (or all) of the data analysis can be performed without guidemaps or workmaps, as may be desired by sophisticated users. This can be done in three distinct ways. One is to use menubar menus (the menubar is always displayed). Another is to type statements in ViDal, ViSta's Data Analysis Language. The third is to use ViDal scripts. Note that we can display the workmap at any time to see the entire history of the analysis, even though we have not been displaying it during the analysis. All of the information necessary to construct the workmap is created as the analysis session unfolds, even when the workmap is hidden. Also note that in any of the above situations we can display the guidemap so that we can use an expert's advice as to how to proceed with the analysis. These aspects of ViSta, which we do not discuss here, are described by Young (1994) and Young & Lubinsky (1994).

5.0 Creating Statistical Strategies

The guidemaps that embody statistical strategy are created while in "authoring" mode. In this mode there is an **Author's WorkBench** window in which new guidemaps are created. In addition, a **Tools** menu is added to the menubar, and the action of all **Data**, **Transform**, **Analyze** and **Model** menu items is enhanced.

Taken together, the modified menu items and the new **Tools** menu items are "guidetools" that are used to create new guidemaps.The expert uses these guidetools to create the buttons that are to become the nodes of the guidemap. Recall that there are flow buttons, which control the flow between portions of the analysis, and procedure buttons, which control the use of data-analysis procedures. The **Tools** menu creates flow buttons, while the other menus create procedure buttons.

Procedure buttons are created by using those menu items that are needed to perform the specific type of data analysis for which guidance is being created. When in authoring mode, the action of the menu items is modified so that, in addition to the analysis action taking place, a button is placed on the author's workbench (the button's title is the same as the menu item's name).

Note the basic design philosophy underlying the creation of statistical strategies: The expert creates the guidemap's data-analysis procedure buttons by using the menu system in exactly the same way that s/he would use it when it is not in authoring mode. Since the system is in authoring mode, buttons appear in the workbench window. Otherwise, everything is the same as when the system is not in authoring mode. This design feature means that the expert is free to perform whatever analysis is desired, using whatever data-procedures are appropriate, without any new authoring "features" standing in the way of the authoring process.

On the other hand, flow buttons, which do not correspond to data-analysis actions, are created by using the new authoring "features" that are represented by items of the **Tools** menu. There is a menu item for each type of flow button, including **Link**, **GoTo**, **Return** and **And** items (for icons shown in previous figures), **AutoLink** and **AutoReturn** items that cause a guidemap to automatically link to another guidemap and to automatically return to the linked-from guidemap, and an **Initial** item to indicate which buttons are to be activated when the guidemap is initially displayed. Thus, while the author does not need to learn any new aspects of the system while creating the procedure steps of the data analysis, new features must be learned to indicate flow control (the actual guidance). In a more complete implementation, many additional flow-control features would be available. The literature on visual programming is particularly relevant (Chang, 1990; Shu, 1988; Rasure & Williams, 1991; Myers, 1990).

Once the expert has placed two or more buttons or icons on the workbench, s/he can connect them together with an arrow drawing tool. Of course, at any time the buttons and icons can be dragged to new locations to give the guidemap a more pleasing and comprehensible layout. The arrows automatically reposition themselves to reflect the new layout. Of course, when the map is entirely created, the expert saves it for later use by the novice. Finally, the expert must create the help information that is displayed when the novice clicks on the ?? side of a button. This is done by using an ordinary text editor, and by saving files with names that meet certain conditions so that they can be found when needed.

6.0 Discussion

In this section we discuss issues related to our research concepts and their implementation. These issues include statistical objects, hypertext and hypercode, visual programming and program visualization, a cognitive model of data analysis, and methods for empirically evaluating our ideas.

6.1 Statistical Objects

ViSta is implemented using the object-oriented programming features of Lisp-Stat (Tierney, 1990). Lisp-Stat has been extremely useful for prototyping and testing our ideas in visual statistics. We doubt that an academic statistician could have prototyped these ideas on his own without monetary support in approximately 18 man-months (as was done here by the first author) using any other system. Indeed, we began prototyping our ideas in S (Becker, Chambers & Wilks, 1988) and after 9 man-months abandoned the effort. The fact that Lisp-Stat includes object, windowing and graphical systems made the effort much easier. One of the great strengths of Lisp-Stat is the openness of the system, particularly of the graphical subsystem. It was fairly straight-forward to develop guidemaps and workmaps, with the system running on a variety of hardware.

One of the strengths of Lisp-Stat is that it supports object-oriented programming, as do New-S (Becker, Chambers & Wilks, 1988) and APL2STAT (Friendly & Fox, 1994). One essential idea of the system is that an object is a collection of data structures and computing methods. Certain objects, called *prototype* objects, have data structures that do not contain data. Prototypes are used to create *instances*, which are objects with the same data structures and methods as the prototypes, where the data structures contain data. Another essential idea is that object prototypes can *inherit* data structures and computing methods from other object prototypes. This greatly simplifies the programming task, since already existing data structures and computing methods do not have to be reprogrammed for new objects.

At the heart of our implementation is a system architecture involving statistical object prototypes. Our system is based on the insight that the fundamental statistical object prototype should be the data prototype, and that programming would be greatly simplified if statistical models were prototype objects inheriting from data object prototypes. Our experience, which suggests that this insight was correct, also suggests that a mixed hierarchical structure based on multiple inheritance would further simplify programming structure.

Our current view of the best system architecture is shown in Figure 4. We now believe that *one* of the most fundamental kinds of statistical object prototypes is the "data prototype". It has data structures and computing methods that are

Insert Figure 4 About Here

needed by all statistical objects. In addition, we believe there should be two *additional* fundamental statistical object prototypes, called the "model prototype" and the "transformation prototype". These have data structures and computing methods that are unique to models or transformations. We further suggest that there should be specialized data object prototypes for specialized kinds of data (multivariate, relational, tabular, etc.). For each kind of specialized data prototype there is a corresponding specialized model object prototype which multiply inherits data structures and computing methods from the generalized model prototype and from the appropriate specialized data prototype. The same type of multiple inheritance applies to the transformation prototypes as well.

We now come to the bottom of the hierarchy. Here we have specific model and transformation prototypes that inherit structures and methods from the appropriate model or transformation prototype. These could include specific multivariate model prototypes for multiple regression, principal components and correspondence analysis, a relational model prototype for multidimensional scaling, and a tabular model prototype for analysis of variance. There could also be a number of specific transformation prototypes, as shown in the figure. (All model and transformation prototypes shown in the figure are implemented in ViSta, though with the simple inheritance structure).

Note the following interesting aspect of this architecture: The analysis *procedures* discussed in previous sections of this paper correspond to what we refer to here as *specific model prototypes,* whereas the *models* that we discussed earlier are *instances* of these specific model prototypes. Thus, models are instances of procedures! This is as it should be — an analysis procedure is a collection of computing methods and data structures waiting to be applied to information, whereas a model is the specific instance of having applied those methods to a specific collection of data. Also, an analysis procedure has an empty data structure, whereas a model has a data structure that has been filled with the results of applying the method to some data. Note also that this simplifies the ability to update analysis sessions with new or revised data: Whenever new data are at hand, all models can be easily updated simply by updating their data and reapplying their methods to the new data.

6.2 Hypertext and Hypercode

Hypertext (or, more generally, hypermedia) is a generic approach to linking and structuring all forms of computerized materials so that nonlinear, dynamic documents can be constructed (for more information, consult Woodhead (1990) or Martin (1990)). Hypermedia consist of nodes that are connected by links. The nodes contain the materials, which may be text, diagrams, animations, images, video, sound, computer programs or any other computerized information. The links provide a mechanism for nonlinear navigation among the nodes. The nodes may be linked together into web, hierarchical, cyclic, or other structures. Hypermedia always have tools for navigating the link structure and for displaying the node material.

Clearly, our help system is a hypertext: The guidemap buttons are the nodes that contain the help text, and the arrows are the links between the nodes. In addition, the ?? side of a guidemap button is the tool that accesses and displays the hypertext. The buttons also navigate the hypertext. Finally, the structure of the hypertext is shown by the structure of the guidemap.

Of much more interest is the fact that our guidance system is a hypercode, a form of hypermedia where the materials are computer programs. Note that the structure of the hypercode is represented by the structure of the guidemap, and that the hypercode is navigated by clicking on the !! side of guidemap buttons. When the novice analyst clicks on the !! side of a button, the button not only navigates to a particular piece of hypercode, but also causes the execution of that piece of code. Thus, from the point-of-view of the novice user, the guidemaps display the structure of the guidance hypercode, provide a means of navigating through it, and a means of executing pieces of it. (Note that the guidemaps also display the structure of the *help* hypertext, provide a means of navigating through it, and for displaying pieces of it. Thus, both the hypertext and hypercode are seamlessly unified.)

It follows that the expert user's process of authoring guidemaps is, in fact, a process for writing hypercode. As described above, authoring involves creating two kinds of buttons: action buttons and flow buttons. When an action button is created, the code that is written is a ViSta function which parallels a data-analysis menu item and which causes a data-analysis step to take place. On the other hand, when the author creates a flow button, the code that is written consists of standard Lisp flow control functions.

Thus, authoring guidemaps is computer programming. However, it is not the usual type of programming in which the programmer types statements. Rather, it is one in which the statements get generated automatically when the author (programmer) selects a button. This form of computer programming is known as visual programming, which is discussed in the next section.

6.3 Visual Programming and Program Visualization

Visual programming and program visualization are very active areas of research in computer science. There goal is to simplify programming, and to make programming accessible to a wider audience. They attempt to reach this goal by combining the disciplines of interactive graphics, computer languages and software engineering to take advantage of a person's nonverbal visual capabilities and a computer's interactive graphical capabilities.

Conventional textual computer languages process program instructions and manipulate information that exist in one-dimensional, nongraphical (textual) streams. Visual programming, by contrast, refers to a way for people to create programs using graphical methods. These icons can be viewed as two-dimensional graphical instructions (Myers, 1990), as opposed to one-dimensional textual instructions (although the two-dimensional visual program is translated into an underlying one-dimensional textual program). Program visualization, on the other hand, is an entirely different concept: Here, the program is specified in the usual textual manner, but is then illustrated visually in some form. Thus, the program is specified as text and translated into graphics. Note that this reverses the process involved in visual programming, where the program is specified as graphics and is translated into text.

Guidemaps and workmaps are simple examples of visual programming and program visualization. Guidemaps are visual programs which have been created by an expert using a visual authoring system, and which are "executed" by the novice. Workmaps are program visualizations which have been created textually (or visually). In fact, when a workmap is saved and re-executed, it becomes a visual program as well as a program visualization.

The earliest visual languages were computerized flowcharts. More recently, visual languages are formally based on graph theory, consisting of nodes and edges (note the connection with hypertext). Often the edges are directed (and called arrows). There are graphs such as “higraphs”, which allow nodes to contain other nodes and which permit arrows to split and join, or “colored petri nets” which allow parallel processing systems to be constructed. A number of visual programming systems use dataflow diagrams. Here the operations are typically put in nodes, and the data flow along the arrows connecting the nodes.

In our work we have based guidemaps on directed cyclic graphs and workmaps on directed acyclic dataflow diagrams (see Young & Smith, 1991). Our developments are limited, however, in that we have not developed looping or conditional branching. Thus, one can argue that our workmaps and guidemaps do not constitute a full visual programming language, since the abstract definition of a computer language requires the inclusion of these capabilities. For this reason, we feel that it would be advantageous to more thoroughly investigate the possibility of developing (or using an existing) visual dataflow language as the basis for a structured graphical interface for performing and guiding data analysis. Two interesting possibilities that bear further examination are VisaVis (Poswig, Vrankar & Morara, 1994) and Khoros (Rasure & Williams, 1991). Both are functional visual programming languages with looping and conditional branching. Khoros is also a dataflow language.

6.4 A Cognitive Model of Data Analysis

Young & Smith (1991) present a structured graphical user interface for data analysis that, like ours, is also designed to improve the data analysis process. One of the cornerstones of their presentation is that the structure of the data-analysis environment should reflect the various modes of cognition used by data analysts during data analysis. This idea leads them to propose a cognitive model for data analysis: There are several specific cognitive modes of behavior and thought used by a data analyst during data analysis, one of these modes being active at any given time, with the activation of a mode depending on the specific kind of ongoing data analysis activity. They then argue that a data-analysis environment should have visible software modes (windows) corresponding to the cognitive modes. They propose three cognitive modes, each being supported by a corresponding software mode: An *exploratory* cognitive mode that is active when a data analyst explores data, and which is supported by a graphical user interface (a system mode) like our guidemap; A *confirmatory* cognitive mode that underlies confirmatory data analysis and is supported by an alphanumeric interface like our language window (discussed in Young & Lubinsky, 1994); A *structure* cognitive mode which is used to construct, maintain and revise a meaningful structure of the overall data-analysis session, and which is supported by a graphical user interface like our workmap.

While we agree with Young & Smith’s assumption that data analysts have different modes of cognition during data analysis, we take the position here that modes of cognition are a function of the level of data-analysis expertise rather than a function of the specific data-analysis activity occurring at a given moment of a data-analysis session. We believe that the same cognitive modes are used by novice, competent, experienced and expert data analysts, but that the distribution of time spent in the various modes changes as the data analyst becomes more experienced with a particular aspect of data analysis.

Thus, we argue that a well-structured data-analysis environment should have software modes (i.e., windows) that reflect the variation in cognitive modes that data analysts employ, and that the frequency of employment of such cognitive (and, therefore, software) modes is a function of the experience of the analyst. We also argue that a specific data analyst does not necessarily always operate at the same level of expertise throughout an entire analysis. Thus, someone may be very experienced with exploring data, but at the same time be less competent when it comes to performing a principal components analysis, and totally unfamiliar with time series analysis. Therefore, we have designed our data-analysis environment to permit an analyst to effortlessly switch between the various interfaces, and to have these interfaces be mutually complementary and seamlessly integrated.

Perhaps the truth of the matter combines Young & Smith's assertion that the user's cognitive mode depends on the specific type of statistical activity that is under way, as well as on our assumption that the user's cognitive mode depends on the level of expertise of the analyst for the specific type of statistical activity that is taking place. In any case, this is a matter for future empirical research, research which could yield improvements in the quality, satisfaction and productivity of the data-analysis process.

6.5 Empirical Evaluation

We have proposed and developed an environment for data analysis that is designed to improve the data analysis process. Our work is based on our own theories about statistical strategy and about how to design an environment based on these theories. While one may (or may not) think that our theories and developments will improve the data analysis process, we have presented no evidence addressing this issue. However, we have presented a formal hypothesis about the effects of our work, and that hypothesis can be empirically evaluated.

As you will recall, our fundamental hypothesis is that data analyses performed by data analysts in an environment that visually guides and structures the analysis will be more productive, accurate, accessible and satisfying than data analysis performed by the same people in an environment without such visual aids.

We can empirically evaluate this hypothesis by designing an experiment in which there are several groups of data analysts (the subjects). All subjects would use ViSta to analyze the same data to obtain answers to the same specific questions. The groups would vary according to what visual aids they were provided to do the analysis: Group ***GS*** (guided and structured) would have both guidemaps and workmaps, whereas group ***G*** would have only guidemaps, and group ***S*** would have only workmaps. In addition, group ***M*** (menus) would perform the analysis using menus but without guidemaps and workmaps, and group ***L*** (language) would perform the analysis by typing statements in ViDal

(ViSta's Data Analysis Language), seeing no menus, guidemaps or workmaps. Our hypothesis leads us to predict that group ***GS*** would perform the best, groups ***L*** and ***M*** the worst, with groups ***G*** and ***S*** in between.

We must, of course, empirically define "productive, accurate, accessible and satisfying". While this is not the place to present these empirical definitions, we can anticipate how such measures would be developed: Productivity could be defined by measures of the total number of steps in the analysis, and the number of dead-ends and false turns made during the analysis (as judged by the subjects themselves as well as by expert judges). Accuracy could be defined not only by whether the subjects "got the right answer", but also by whether the techniques used to answer the data analysis question were appropriate (as judged by expert judges). Accessibility and satisfaction could be based on the subject's judgments about how satisfied or frustrated they felt during the analysis, whether they would want to use the environment again, and whether they would recommend it to a friend.

In addition to our basic hypothesis, we hypothesize that the amount of improvement of the data analysis process should be greatest for novices, with less improvement for subjects with greater statistical sophistication. To test this hypothesis our experiment would have to be repeated with subjects selected from populations with different levels of data analysis sophistication. For example, the novice population could be students in classes that provide the first introduction to data analysis. An intermediate population could be graduate students who have taken several data analysis courses. An advanced population could be professional data analysts.

Clearly, when software is developed whose purpose is to improve the user's experience, that experience should be empirically evaluated. Unfortunately, this software development step is time-consuming and expensive, and is rarely taken. However, we feel that our research and development effort is incomplete unless we know whether our ideas really work. Thus, we look forward, with some trepidation, to empirically evaluating the ViSta user's experience.

7.0 Conclusion

Understanding and representing statistical strategy is a relatively new area of research that is just now gaining momentum. As the capability of computers continues to increase, while their price continues to decrease, the audience for complex software systems such as data-analysis systems will become wider and more naive. Thus, it is imperative that these systems be designed to guide data analysts who need the guidance, while at the same time be able to provide full data-analysis power. An efficacious way of doing this is certainly needed, and we believe that visual statistical strategies should be available to guide and structure the data-analysis process so that relatively novice users can perform high-quality data analyses.

Naturally, we hope that our methods for guiding novice data analysts will prove useful. Of much greater importance, however, is our basic point: Concentrated attention should be given by computational statisticians to the development of software environments which improve the data analyst's experience. We hope that our work is a useful first step in what we see as an important but neglected area of computational statistics.

8.0 References

1. Baxter, R. & Cameron, M. (1991) Comment on Lisp-Stat. *Statistical Science*, **6**, 339-343.
2. BBN Software (1989) *RS/Explore MULREG Reference Manual*. BBN Software Products Corp., Cambridge, Mass.
3. BBN Software (1994) *Cornerstone*. BBN Software Products Corp., Cambridge, Mass.
4. Becker, R.A., Chambers, J.M. & Wilks, A.R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.
5. Chambers, J.M. (1981) Some thoughts on expert software. *Computing Science & Statistics*, **13**, 36-40.
6. Chang, Shi-Kuo (1990) *Principles of Visual Programming systems*. Prentice-Hall, Englewood Cliffs, NJ.
7. Cook, R.D. & Weisberg, S. (1994) *An Introduction to Regression Graphics.* Wiley, New York.
8. Daniel, C. and Wood, F.S. (1980) *Fitting Equations to Data*. John Wiley and Sons.
9. Friendly, M & Fox, J. (1994) Using APL2 to Create an Object-Oriented Environment for Statistical Computation. *J. Computation and Graphical Statistics*, **4**, 387-408.
10. Gale, W.A. (1988) *Artificial Intelligence and Statistics*. Addison Wesley, Reading, Massachusetts.
11. Gale, W.A., Hand, D.J. & Kelly, A.E. (1993) Statistical Applications of Artificial Intelligence. In: C.R. Rao, *Handbook of Statistics: Computational Statistics*, Amsterdam: Elsevier North-Holland, **9**, 535-576.
12. Gale, W.A. & Pregibon, D. (1982) An expert System for Regression Analysis. *Computing Science & Statistics*, **14**, 110-117.
13. Gorsuch, R.L.(1983) *Factor Analysis*. Lawrence Erlbaum Associates, Hillsdale, NJ.
14. Hand, D.J. (1984) Statistical Expert Systems: Design, *The Statistician*, **33**, 351-369.
15. Hand, D.J. (1985) Statistical Expert Systems: Necessary Attributes, *Journal of Applied Stat*., **12**, 19-27.
16. Hand, D.J. (1993) *Artificial Intelligence Frontiers in Statistics.* London: Chapman and Hall.
17. Hand, D.J. (1994) Statistical Expert Systems, *Chance*, **7**, 28-31,34.
18. Lubinsky, D.J. & Pregibon, D. (1988) Data Analysis as Search. *Journal of Econometrics,* **38**, 247-268.
19. Lubinsky, D.J. (1991) Comment on Lisp-Stat. *Statistical Science*, **6**, 352-360.
20. Martin, J. (1990) *Hyperdocuments and how to create them*. Prentice Hall, Englewood Cliffs, NJ
21. Myers, B.A. (1990) Taxonomies of Visual Programming and Program Visualization. *Journal of Visual Languages and Computing*, **1**, 97-123.
22. Nelder, J.A. (1977) Intelligent Programs, the Next Stage in Statistical Computing. In Barra (Ed.) *Recent Developments in Statistics*. North-Holland, Amsterdam, 79-86.

23. Norusis, Marija J. (1990) SPSS Base System User's Guide. SPSS Inc., Chicago, Illinois

24. Oldford, W. & Peters, S. (1988) DINDE: Towards more Sophisticated Software Environments for Statistics. *Siam Journal of Scientific and Statistical Computing*, **9**, 191-211.

25. Poswig, J., Vrankar, G. & Morara, C. (1994) VisaVis: A Higher-order Functional Visual Programming Language. *Journal of Visual Languages and Computing*, **5**, 83-111.

26. Pregibon, D. & Gale, W.P. (1984) REX: An expert system for regression analysis. *Proc. COMPSTAT* **84**, 242-248.

27. Rasure, J.R. & Williams, C.S. (1991) An Integrated Data Flow Visual Language and Software Development Environment. *Journal of Visual Languages and Computing*, **2**, 217-246.

28. SAS Institute (1990) *SAS User's Guide*. SAS Institute, Inc., Cary, NC.

29. SAS Institute (1990) *JMP User's Guide*. SAS Institute, Inc., Cary, NC.

30. Shu, Nan C. (1988) Visual Programing. Van Nostrad Reinhold. New York.

31. Thisted, R.A. (1988) Representing Statistical Knowledge for Expert Date Analysis Systems,. In Gale, W.P. (Ed.) *Artificial Intelligence and Statistics*. Addison Wesley, Reading, Massachusetts.

32. Tierney, L. (1990) *Lisp-Stat: An Object-Oriented Environment for Statistical Computing & Dynamic Graphics*. Addison-Wesley, Reading, Massachusetts.

33. Van den Berg, G.M. (1992) *Choosing an Analysis Method: An Empirical Study of Statisticians' Ideas in View of the Design of Computerized Support.* DSWO Press, Leiden, The Netherlands.

34. Velleman, P.F. & Velleman, A.Y. (1988) *Data Desk Professional*. Odesta Corp., Northbrook, IL.

35. Weihs, C. (1991) Comment on Lisp-Stat. *Statistical Science*, **6**, 344-348.

36. Woodhead, N. (1990) *Hypertext & Hypermedia: Theory and Applications*. Addison-Wesley. New York.

37. Young, F.W. (1991) Comment on Lisp-Stat. *Statistical Science*, **6**, 349-352.

38. Young, F.W. (1994) ViSta - The Visual Statistics System. Research Memorandum # 94-1, L.L. Thurstone Psychometric Laboratory, Univ. N. Carolina, Chapel Hill NC.

39. Young, F.W. & Lubinsky, D.J. (1994) Guided Data Analysis: On the Representation, Use and Creation of Visual Statistical Strategies. Research Report # 94-1, L.L. Thurstone Psychometric Laboratory, Univ. N. Carolina, Chapel Hill NC.

40. Young, F.W. and Smith, J.B. (1991) Towards a Structured Data Analysis Environment: A Cognition-Based Design. In: Buja, A. & Tukey, P.A. (Eds.) Computing and Graphics in Statistics, 36, 253-279. New York: Springer-Verlag.

9.0 Figures

Figure Captions

FIGURE 1. Formal Representation of Statistical Strategy in the WorkMap and GuideMap

FIGURE 2. Formal Representation of Strategy for Exploring Data

FIGURE 3. Strategy for Exploring Data after using several analysis procedures

FIGURE 4. Statistical Object Prototype Structure

FIGURE 1.

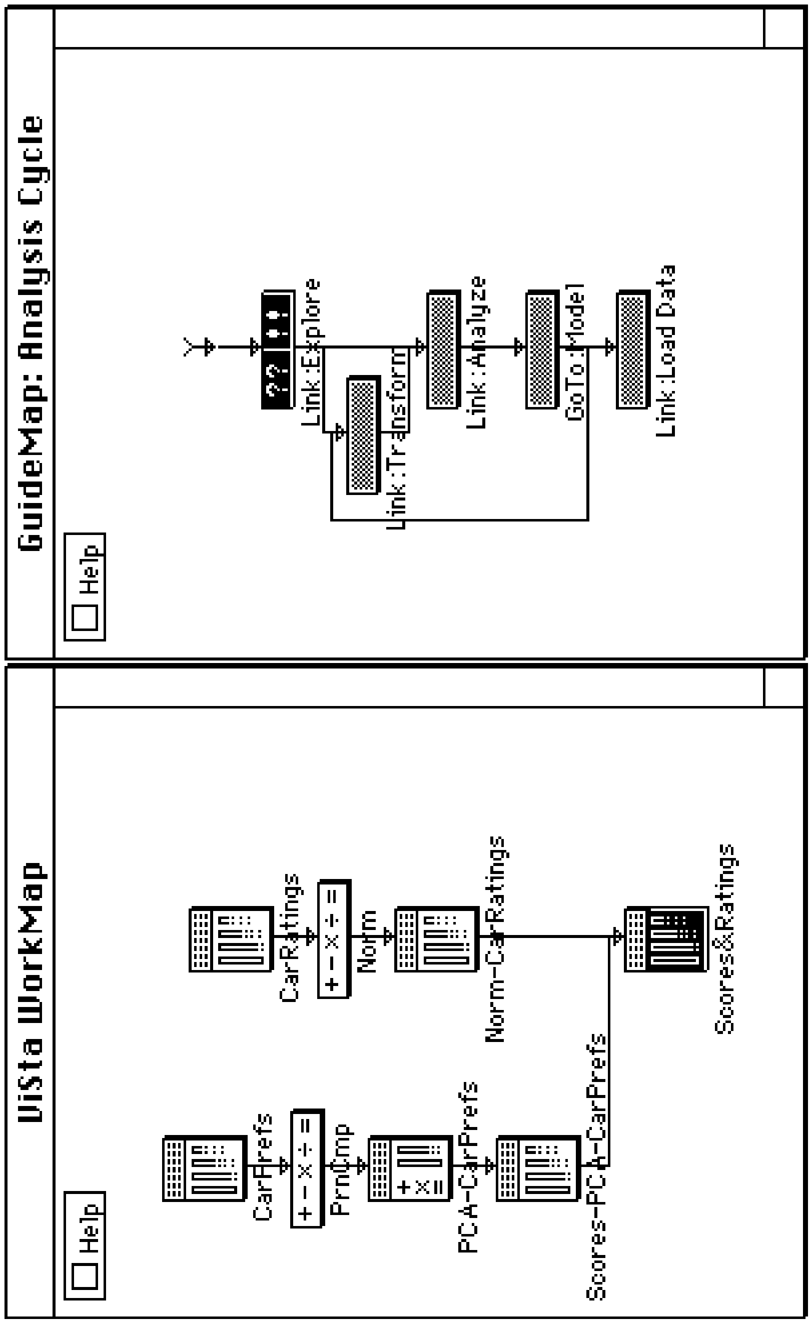

GuideMap: Analysis Cycle
Help
Link:Explore
Link:Transform
Link:Analyze
GoTo:Model
Link:Load Data
ViSta WorkMap
Help
CarPrefs
PrnCmp
PCA-CarPrefs
Scores-PCA-CarPrefs
CarRatings
Norm
Norm-CarRatings
Scores&Ratings

FIGURE 2.

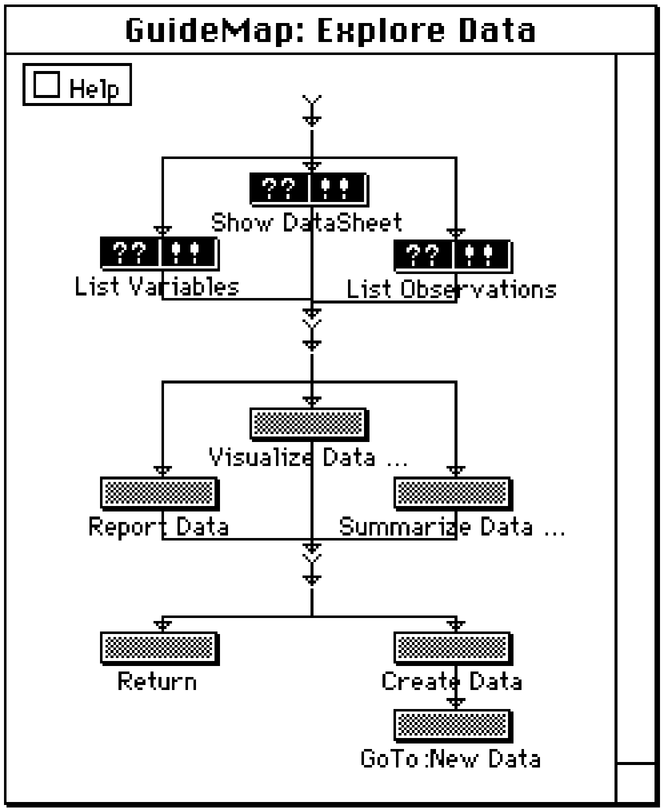

GuideMap: Explore Data
Help
?? !!
Show DataSheet
List Variables
List Observations
Visualize Data ...
Report Data
Summarize Data ...
Return
Create Data
GoTo :New Data

FIGURE 3.

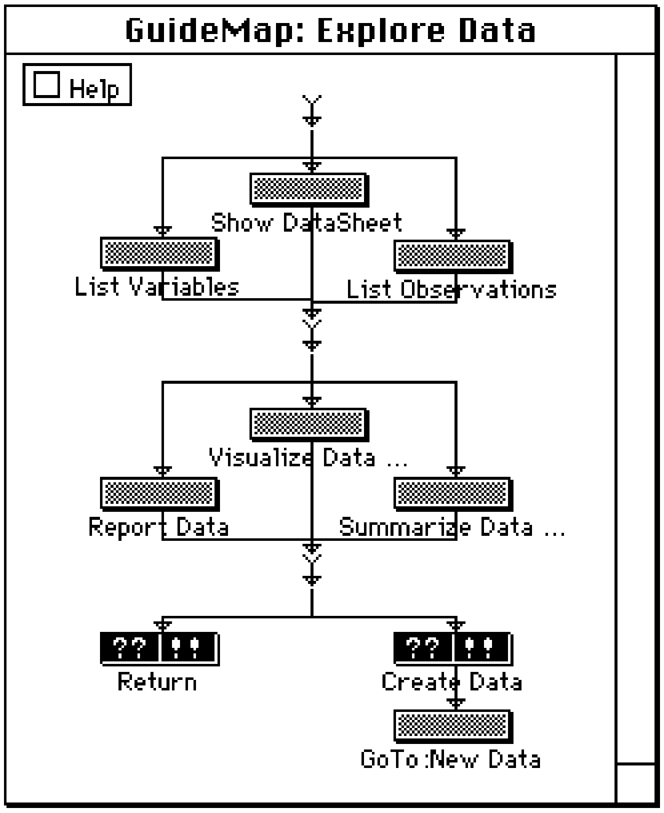
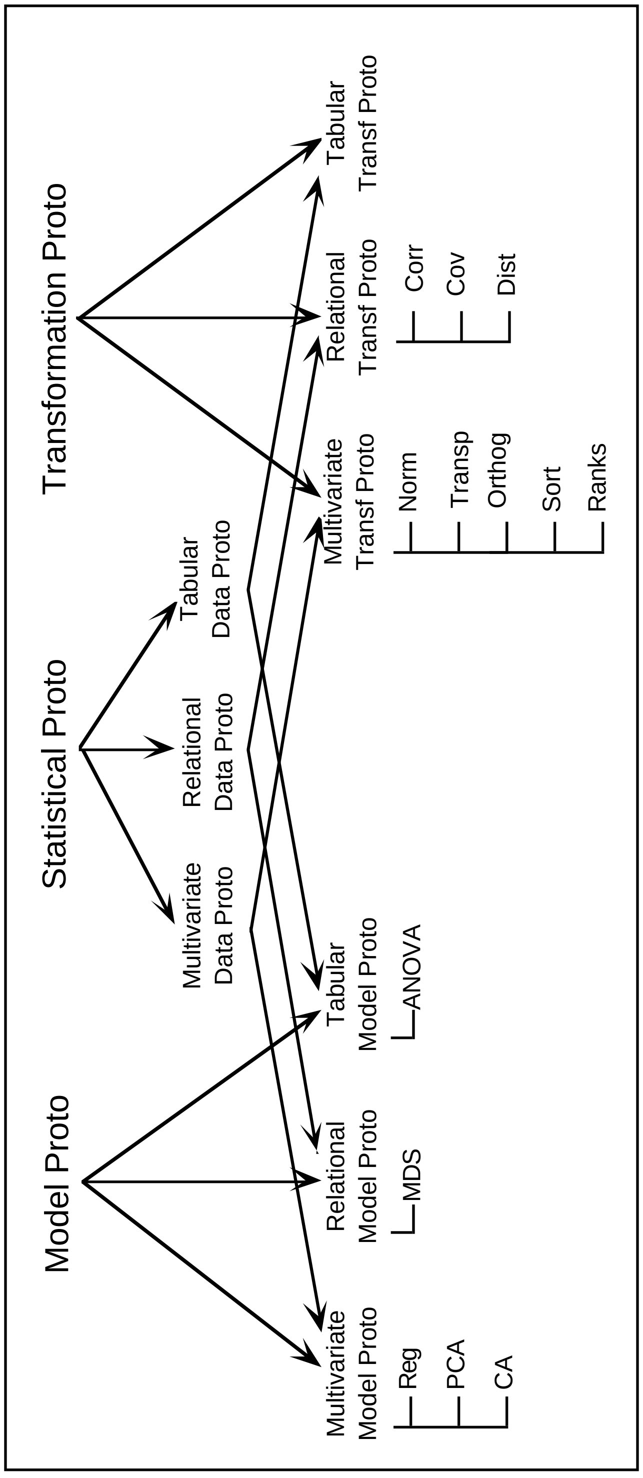

GuideMap: Explore Data
Help
Show DataSheet
List Variables
List Observations
Visualize Data ...
Report Data
Summarize Data ...
?? !!
Return
?? !!
Create Data
GoTo :New Data

FIGURE 4.


Model Proto
Statistical Proto
Transformation Proto
Multivariate Model Proto
Relational Model Proto
Tabular Model Proto
Multivariate Data Proto
Relational Data Proto
Tabular Data Proto
Multivariate Transf Proto
Relational Transf Proto
Tabular Transf Proto
Reg
PCA
CA
MDS
ANOVA
Norm
Transp
Orthog
Sort
Ranks
Corr
Cov
Dist