Learning from Data by Guiding the Analyst: On the Representation, Use and Creation of Visual Statistical Strategies

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 analysts so that they can learn about the structure of their data. 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.

The interface consists of two interacting windows: the *guidemap* and the *workmap*. An example is shown in Figure 1 (a screen image from **ViSta** (Young, 1994), software that implements the ideas in this paper).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 show which statistical object is the focus of the strategy.

1.0 Background

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

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 on-going 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.

Our formal representation of session structure is the workmap: 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 chronological sequence of the objects and procedures (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 on-going data-analysis session, can be selected and reviewed at any time. The workmap visualizes the history of the on-going data analysis.

Guiding Data Analysis: At each step of a data analysis 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 improves when it occurs in an environment that guides the actions taken by the analyst to understand the data.

We use the Artificial Intelligence (AI) notion of strategy as a basis for developing methods for guiding data analysts. Several statisticians have developed the notion of a statistical strategy (as reviewed by Gale, Hand & Kelly, 1993). In our 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. In our 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.

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.

2.0 Representing Statistical Strategy

Our definition of statistical strategy involves a *formal representation.*. Our formal representation consists of graph structures like that shown in the guidemap window on the right side of Figure 1. The guidemap, titled **Analysis Cycle**, presents the overall statistical strategy. This specific guidemap is always the first guidemap for a newly created multivariate dataset object. It is only a small portion of the overall strategy, since additional "sub"-guidemaps can be displayed by clicking on the buttons. For example, clicking on the Link:Explore button causes the guidemap in Figure 2 to appear. Taken as a whole, the set of guidemaps are our formal representation of statistical strategy.

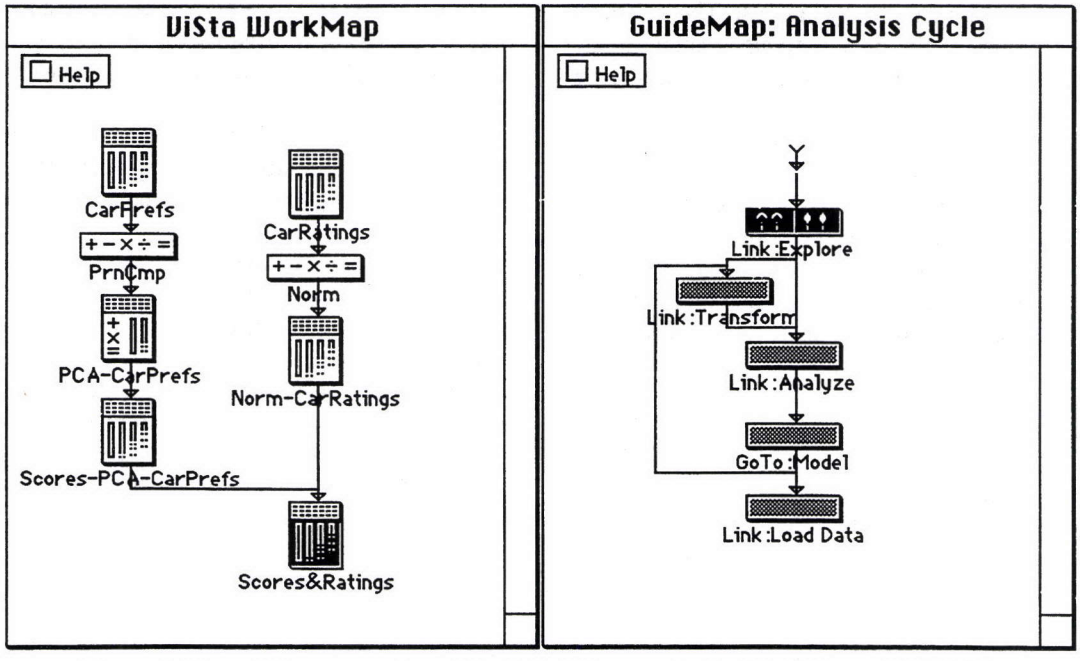

Figure 1: Formal Representation of Statistical Strategy in the WorkMap and GuideMap

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. In Figure 1, the focus object is the "Scores&Ratings" dataset. The workmap itself shows where this object fits into the structure of the overall on-going analysis. The two separate windows emphasize the separation between the on-going data analysis (mapped in the workmap) and the strategy guiding the data analysis (mapped in the guidemap).

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. It may do its own computation, or may 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.

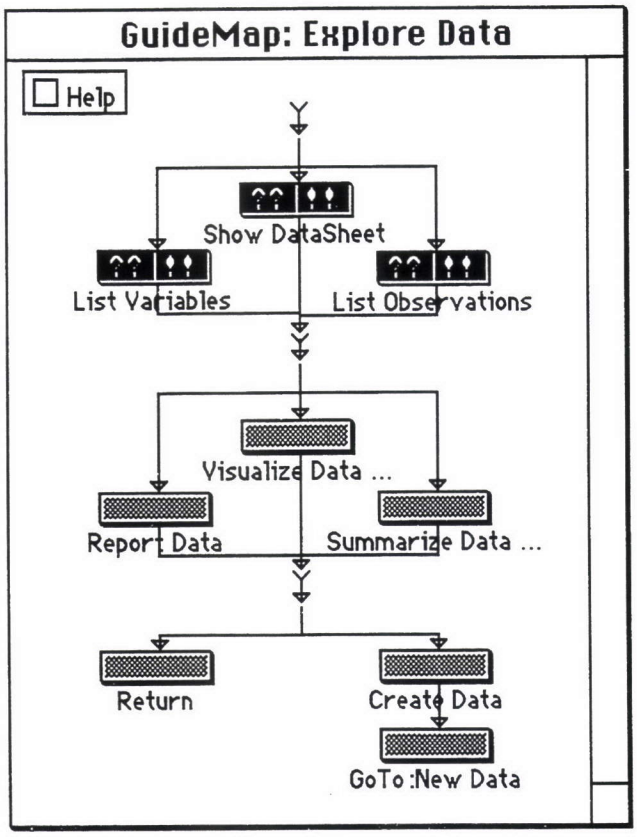

Figure 2: Formal Representation of Strategy for Exploring Data

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. In addition, some Flow Buttons represent represent entire portions of the guidemap. These are called Macro Buttons.

Flow buttons include the Link, GoTo and Return buttons shown in the figures. These buttons take the user to other guidemaps. The Link button takes the analyst to a new strategy, whereas the Return button returns to the linked-from strategy. The Link button is 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 (when it is highlighted) 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 de-activate, 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. All buttons other than flow buttons are procedure buttons that activate data-analysis procedures.

The evolving progress of the data-analysis session is shown 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 and the flow of data into or out of a data-analysis object or procedure.

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).

We turn now to the workmap shown on the left side of 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. This dataset is the current focus of the statistical strategy, as is indicated by its highlighting.

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 from a strategy reflects the fact that a strategy is cyclic. However, analysts can quit a strategy whenever they choose by using the window's close box.

3.0 Using Statistical Strategies

Lets consider how the guidemap in Figure 1 works. First of all, note that all of the buttons in the guidemap in Figure 1 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.

Since Link:Explore button is a *macro* button, 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 a 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: All of the buttons that are connected to an *and* icon must be used before guidance can flow through it to the buttons that follow it.

Note how the strategy has guided the analyst: As shown in Figure 1, the analyst must explore the data first, and must analyze the data before inspecting the model. In Figure 2 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. In Figure 2 it is possible to visualize the data before seeing summary statistics, or to reverse this order.

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. Entire data analyses can be created from the workmap without ever seeing a guidemap. This is done by clicking the mouse on the body of an icon to obtain a pop-up menu of actions that the icon supports (the upper portion of the icon is used to drag the icon, or the whole tree below the icon, to a new location). These menu-items are also accessible from menubar menus. Analysis procedures are accessed from an optional toolbar (that can be shown in the workmap) or from the menubar.

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 with conventional systems which do not keep a full history of a data analysis session. This can be done across sessions by saving (portions of) the workmap and reloading it in a later session.

Also, note that when the data analyst is performing the analysis directly from the workmap, 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 otherwise.

Finally, portions (or all) of the data analysis can be performed with neither guidemaps nor 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 the underlying data-analysis language. The third is to use 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.

4.0 Creating Statistical Strategies

The guidemaps that embody statistical strategy are created while in "authoring" mode, a mode which provides the author with visual programming tools used to create the guidemaps. In this mode there is an **Author's Work-Bench** window in which new guidemaps are created. In addition, a **Tools** menu is added to the menubar, and the action of all menu items is enhanced. The enhanced menu items and the **Tools** menu items become "guidetools" that the expert uses to create guidemaps.

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 enhanced 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".

On the other hand, flow buttons, which do not correspond to data-analysis actions, are created by using the **Tools** menu, requiring that the expert use new authoring features. There is a menu item for each type of flow button shown in previous figures, as well as items that cause a guidemap to automatically link to or return from another guidemap, and an item to indicate which buttons are initially activated. Buttons and icons can be connected together with an arrow drawing tool and can be dragged to new locations to give the guidemap a pleasing layout. Finally, the expert creates the help information that is displayed by the ?? side of a button with an ordinary text editor.

5.0 System Architecture

In this paper we have discussed our concepts concerning *representing*, *using* and *creating* statistical strategies, We have also presented a structured graphical user interface which implements our concepts. In this section we discuss the architecture of our implementation, focusing on the architecture for representing statistical strategy. See Young & Lubinsky (1994) for a discussion of the architecture for using and creating statistical strategies.

Our implementation, which we call **ViSta** (for Visual Statistics), is written in Lisp using the Lisp-Stat system for statistical computation and dynamic graphics (Tierney, 1990). **ViSta** adds our structured graphical user interface to Lisp-Stat's statistical engine, object system, graphical system and windowing environment. **ViSta** uses the Lisp-Stat object-oriented programming system to implement the interface.

The *representation* of our concepts about statistical strategy (i.e., the interface) is built on a foundation of three interacting object families. These families are statistical objects (data, model and transformation objects); map window objects (workmaps and guidemaps); and icon objects (data, model, procedure and guide icons). While the statistical object system is invisible to the user, it is the foundation upon which workmaps and

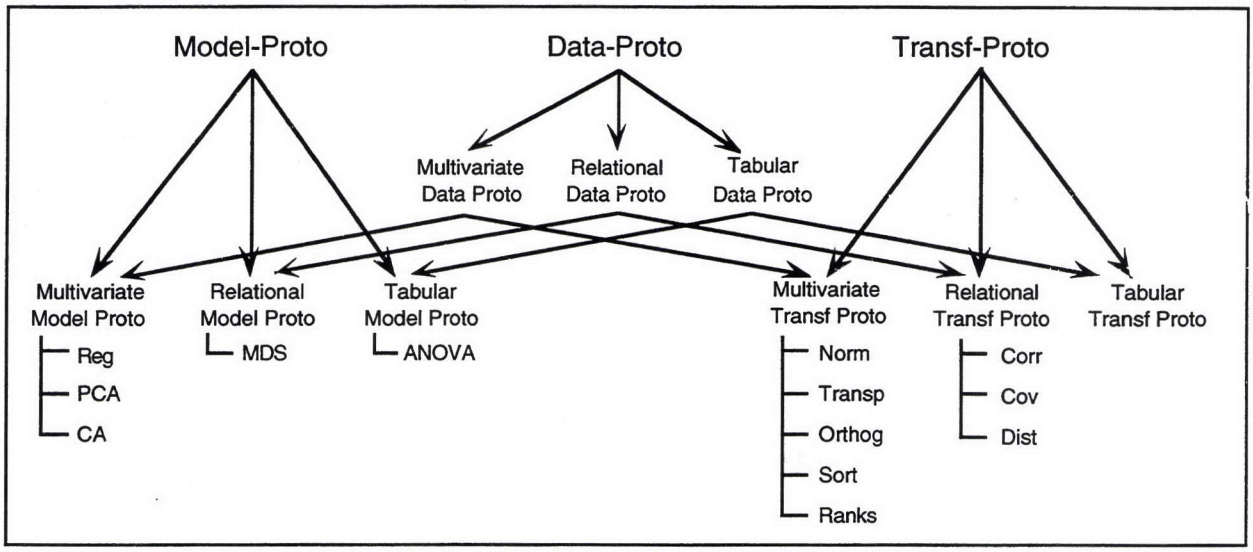

Figure 3: Statistical Object Prototype Structure

guidemaps are built. We discuss the three object families in the remainder of this section.

Statistical Objects: At the heart of the ViSta architecture for representing statistical strategy is a system of statistical object prototypes. These prototypes are organized as a mixed hierarchy, as shown in Figure 3[1]. The most fundamental statistical object prototype is Data-Proto. It contains information and methods that are needed by all statistical objects. The information includes data, the object's title, the object identification of the associated icon object, the parents and children of the object (in the sense of which other objects are above or below in the workmap and guidemap), whether the object needs computing, etc. The methods include methods for storing the information about the object, for creating the object's icon and connecting lines in the workmap, etc. The other two main classes of statistical objects are Model-Proto, which has information and methods for the dialog boxes that obtain model options, and Transf-Proto, which has an extra information slot for the transformed data.

Inheriting from Data-Proto are three specialized data prototypes for multivariate, relational and tabular data. These three prototypes have information and methods that are unique to their type of data. For example, the multivariate prototype has information and methods for the number of rows and columns, whereas the tabular prototype has information and methods concerning the number of ways and the number of levels of each way.

Inheriting from both the Model-Proto and the Data-Proto are three kinds of model prototypes designed for modeling multivariate, relational and tabular data. These prototypes multiply inherit from their specific type of data as well as from model-proto since they need the information and methods of both. The same type of multiple inheritance applies to the three kinds of transformation prototypes as well.

Finally, inheriting from each of the three model prototypes are prototypes for specific types of models. At the time this is being written, these currently include multivariate model prototypes for multivariate multiple regression analysis, principal component analysis and correspondence analysis, a relational model prototype for multidimensional scaling, and a tabular model prototype for analysis of variance. There are also the specific transformation prototypes shown in Figure 3.

In previous sections of this paper we discussed data and model objects. These are specific instances of one of the data or model prototypes. Our discussion used examples of MV-Data-Proto (for multivariate data) and PCA-Model-Proto (for a principal components model). The icons for these models are shown in Figure 1.

Note the following interesting aspect of our architecture: The analysis *procedures* discussed in previous sections of the paper correspond to what we refer to here as

1. The structure that exists at the time of this writing is a simple hierarchy. It is being modified to the mixed hierarchical structure described in this section. The mixed hierarchical structure, based on multiple inheritance, simplifies several programming aspects.

model prototypes, whereas the *models* that we discussed earlier are *instances* of model prototypes. Thus, models are instances of procedures! This is as it should be — an analysis procedure is a collection of methods (in Lisp terminology) 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 empty information slots (in Lisp terminology), whereas a model has information slots that have been filled with the results of applying the methods to some data. Note also that this simplifies updating analysis sessions with new or revised data: Whenever new data are at hand, all models can be easily updated simply by updating their data information and reapplying their methods.

Map Objects: The workmap and guidemap object structure is quite simple: It consists of a iconmap-proto from which inherits the workmap-proto, from which inherits the guidemap-proto. The iconmap-proto contains all of the information-slots and methods that are needed by all icon maps. These include, for example, slots and methods for the number of icons, the entire list of icons, a list of connections between icons, the identification of which icon is highlighted, etc. There are also methods for drawing the map, for dealing with mouse-clicks, for adding new icons, etc. The workmap-proto has information-slots and methods for keeping track of each type of icon (defined below) for updating menus (which maintain lists of data and model objects), operating the toolbar, etc. The iconmap-proto has its own methods for redrawing and dealing with mouse-clicks, for the actions of icons, for storing old guidemaps for later use, and for reading, creating and linking to new guidemaps.

Icon Objects: The icon structure is also very simple: There is a basic icon-proto which has information-slots and methods for icon bitmaps, location, titles. Inheriting from icon-proto are seven specific icon prototypes one for each of the three kinds of data objects, one each for procedures, models, guide-buttons, and and-icons. Each of these has its own bitmap. Some have specialized methods that are appropriate to the specific icon.

6.0 Discussion

In this section we discuss the relation of our work to concepts with their origins in computer and cognitive science: hypertext, visual programming, and cognitive processing.

6.1 Hypertext and Hypercode

Hypertext (or, more generally, hypermedia) is a generic approach to linking and structuring all forms of computerized materials so that non-linear, 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 non-linear 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 naive 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 naive 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.

6.2 Visual Programming & 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 non-verbal visual capabilities and a computer's interactive graphical capabilities.

Conventional textual computer languages process program instructions that exists in nongraphical (textual) streams. Visual programming, by contrast, refers to a way for people to create programs using graphical methods (although the visual program is translated into a 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.

We have based guidemaps on directed cyclic graphs and workmaps on directed acyclic dataflow diagrams (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.

We recommend investigating 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 existing systems 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.

6.3 Cognitive Processes and 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 that the data-analysis environment should have visible system modes (windows) which correspond to the cognitive modes assumed to be active in the data analyst during the process of analyzing data.

Their fundamental assumption is that data analysts adopt different *cognitive modes* of behavior and thought at different stages of the data-analysis process. They propose three cognitive modes: An *exploratory* mode that is active when a data analyst explores data, and which is supported by graphical user interface like our guidemap; A *confirmatory* mode that underlies confirmatory data analysis and is supported by an alphanumeric interface like our language window (see Young & Lubinsky, 1994); A *structure* 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 have taken the position 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. Our informal observations lead us to 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 system 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, system) 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.

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.

7.0 Conclusion

Understanding and representing statistical strategy is a relatively new area of research that is just now gaining momentum. Within this area of research, it appears that our visual approach to statistical strategy is new and unique, and is firmly based on current computer science thinking. 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 our visualized statistical strategies have the potential for great payoff in the improvement of the quality, satisfaction and productivity of statistical data analysis.

Naturally, we hope that our visual methods for guiding naive data analysts by visually representing, using and creating statistical strategies will prove useful. Of much greater importance, however, is our basic point: Concentrated attention should be given by computational statisticians to the representation, usage and creation of statistical strategies. We believe that such strategies should be available to guide and structure the data-analysis process so that relatively naive users can perform high-quality data analyses. And we believe that guidance systems should be empirically tested to see if they deliver on their promise.

8.0 References

1. 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.
2. Lubinsky, D.J. & Pregibon, D. (1988) Data Analysis as Search. *Journal of Econometrics*, **38**, 247-268.
3. Martin, J. (1990) *Hyperdocuments and how to create them.* Prentice Hall, Englewood Cliffs, NJ
4. Myers, B.A. (1990) Taxonomies of Visual Programming and Program Visualization. *Journal of Visual Languages and Computing*, **1**, 97-123.
5. Poswig, J., Vrankar, G. & Morara, C. (1994) VisaVis: A Higher-order Functional Visual Programming Language. *Journal of Visual Languages and Computing*, **5**, 83-111.
6. 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.
7. Tierney, L. (1990) *Lisp-Stat: An Object-Oriented Environment for Statistical Computing & Dynamic Graphics*. Addison-Wesley, Reading, Massachusetts.
8. Woodhead, N. (1990) *Hypertext & Hypermedia: Theory and Applications*. Addison-Wesley. New York.
9. Young, F.W. (1994) ViSta - The Visual Statistics System. *Psychometric Lab Memorandum 94-1.* UNC Psychometrics Lab, Chapel Hill, NC.
10. Young, F.W. & Lubinsky, D.J. (1994) Guided Data Analysis: On the Representation, Use and Creation of Visual Statistical Strategies. *Psychometric Lab Report 94-1.* UNC Psychometrics Lab, Chapel Hill, NC.
11. 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.