1986

Data analysis networks in DINDE

R.W. Oldford and S.C. Peters
Massachusetts Institute of Technology

Abstract

DINDE is a highly interactive display oriented system where the user carries out data analysis by building and maintaining a network representation of it. The network links statistically meaningful objects (e.g. scatterplots, regression results, etc.) and is displayed in a mouse-sensitive window. We describe this network model for a statistical analysis and its implementation in DINDE.

1.0 Introduction

A high resolution bit-mapped display, together with a pointing device like a mouse, allows interaction with elements in the display - interactive statistical graphics, for example, have made effective use of this technology. In an experimental statistical analysis system, called DINDE, we explore how the analysis itself might be presented visually, and how the professional analyst might interact with such a display.

We have settled on a directed network representation for the analysis. In this scheme, each node in the network represents a decision point, or step, in the analysis. Links between the nodes are arranged to display logical connections between steps in the analysis. We discuss this representation by way of an example in Section 2.

Section 3 describes how we propose to manage complicated analyses by selectively displaying different parts of the overall analysis network. These displays are called **AnalysisMaps** in DINDE. In Section 4, we discuss the meaning of the links joining nodes in an analysis network. In particular, distinctions are made between links which are causal and those which are merely associative. This leads to views of different aspects of the analysis network, namely, **CausalMaps** and **DataFlowMaps**. Together with **AnalysisMaps**, these are specializations of a generic network display which we call a **View**. Each of these is discussed in Section 5. Finally, some concluding remarks are made in the last section.

The development of DINDE has taken place within a powerful programming environment, namely, a Xerox 1109 personal workstation running InterLisp-D and an object-oriented programming language called LOOPS (see Teitelman and Masinter [1981], Sheil [1983], and Stefik and Bobrow [1985] for more on this environment). The combination of high interaction, extensive graphics, powerful dedicated computing, and powerful programming tools available in this environment has proved to be enormous leverage for building *and* designing DINDE.

2.0 The analysis as a directed network

A simple non-trivial model of an analysis is a tree, where each branch indicates a logical connection between one step of an analysis and another step which follows it. Usually the second step is a direct consequence of an action taken at the first. Each step, or decision point, in the analysis is a place of possible branching. And, since one often returns to some decision point to pursue a different course of action, it is possible that more than one branch will come out of any step.

A little reflection, however, shows that a tree model falls short. Suppose that two branches of the tree represent two different minor analyses that are pursued in parallel. It may happen that a new tack is taken in the analysis based on the combined results of both independent sub-analyses. How should the beginning of this new sub-analysis be shown? The obvious answer is to attach its beginning to the end of both of the previous sub-analyses, thus leading us to describe the whole analysis as an acyclic directed network rather than a tree.

Figure 1 shows the data analysis network associated with a linear regression analysis of the average brain weights on the average body weights of 62 mammals. The direction of the links is from the top of the display down. The figure represents a completed analysis, we next describe how the analysis, and hence the graph, evolved.

The purpose of the analysis is to fit the brain weights as a function of the body weights. At the top of figure 1, the analysis begins with two **FloatVectors** of 62 elements each, called **BodyWeights** and **BrainWeights**, that contain the data.

After constructing the **FloatVectors**, a histogram of the elements of each vector was examined. These histograms showed markedly skewed data. Nevertheless, it was decided to fit a regression model of **BrainWeights** on **BodyWeights**.

This decision is reflected in the analysis network by the node **BivariateRegression**. Note that this node does not include the results of any particular regression fit. Rather, it represents the decision to undertake a regression fit. As we soon describe, actions that are likely to be helpful in pursuing this goal are associated with **BivariateRegression**. These include fitting a straight line of y to x, by either least-squaresor the resistant-line fitting procedure, fitting a smooth curve using a locally linear smoother, and plotting the data in a number of ways.

With the goal of regression in mind, first a simple scatterplot of the points was produced. Perhaps due to the skewness of the data, a resistant line was fitted to the points. The residuals were then examined plotted against the fit.

At this point, the direction of the analysis changed. It was decided to transform the data by taking natural logarithms of each data vector - giving **LnBodyWts** and **LnBrainWts**. From a scatterplot of the transformed data, a straight line was fitted using least-squares. Finally, after a number of residual plots were examined, the analysis was done.

The analysis network is continually displayed and is updated with each action taken. The user literally builds the analysis, using the mouse to select objects in the display and to choose actions from menus at each decision point.

An action is taken at any node by first selecting the node with the mouse. This causes a menu to pop up over the node. Each item in the menu is an action which can be taken at that node; selecting one with the mouse causes it to be performed.

Two different kinds of menus can appear at every node - one with one mouse button depressed during the selection, and the other with a different mouse button depressed. The first menu presents the actions, or decisions, which may result in a new node. The second kind provides the user with the opportunity to access or store information on that node (e.g. storing arbitrary textual information as **Notes** on that node).

The first menu is different for each different kind of node (e.g. allowing "*LOG*" for **FloatVectors**, and "*Fit the Resistant-Line*" for **ScatterPlots**). Items selected from this menu can cause new analysis nodes to appear. For example, from the **ScatterPlot** titled **PlotOfLogs** the analyst selected the menu item "*Fit a least-squares line*". This produced the **BivariateLeastSquares** object, did the calculations, stored the results on the new object, and, finally, established a link between the two objects - from **PlotOfLogs ScatterPlot** to the new **BivariateLeastSquares** (unnamed). Similarly, selecting the "*LOG*" menu item associated with the **FloatVectors BrainWeights** and **BodyWeights** produced two new **FloatVector** objects, containing the logarithms, and made a link between the appropriate vectors.

Reprinted from the 1986 Proceedings of the Statistical Computing Section of the American Statistical Association

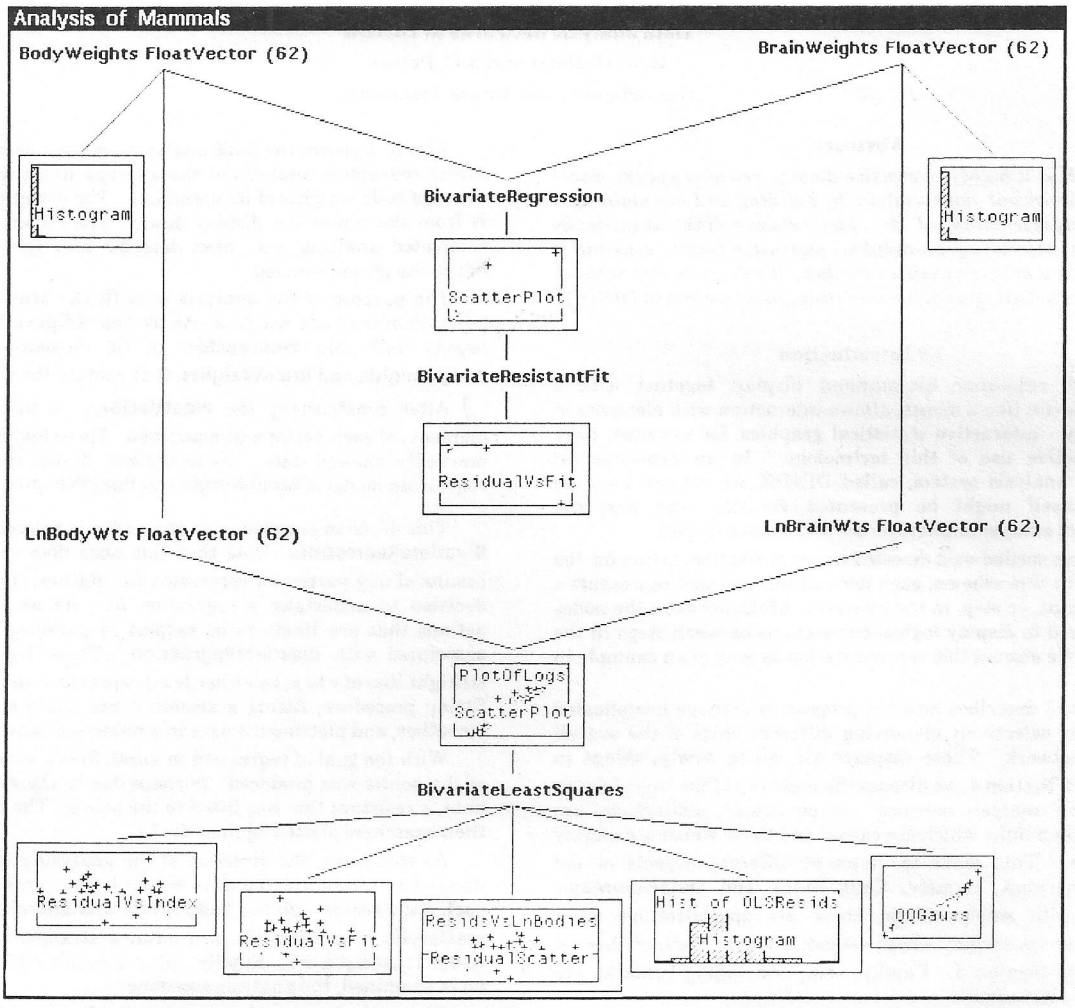

Figure 1. Analysis of mammalian brain and body weights.

The second kind of menu only allows interaction with information on the selected node. It is identical for all nodes in the analysis network and cannot produce any new analysis node. One item from this menu, called *Zoom*, is particularly important. All analysis nodes can be "*Zoom*ed" in on to reveal their hidden detail. *Zoom*ing in on a **ScatterPlot** causes the scatterplot that it represents to appear on the screen; *Zoom*ing in on a **BivariateLeastSquares** node shows its internal components, like t-statistics, coefficient estimates, residual and fitted **FloatVector**s, and so on; *Zoom*ing in on a **FloatVector** displays the actual numbers it contains. *Zoom* is used uniformly to access greater detail on the selected node. In this way, the detailed information is localised and suppressed from the overall picture of the analysis.

New analysis nodes appear in the network in one of three ways: as the result of an action taken from a displayed node (as described above), as the result of evaluating a user-typed expression, or, by selecting a template of the desired kind of node from another display called a toolbox. (More detail on this step by step construction of a data analysis can be found in Oldford and Peters [1985, 1986].) Allowing new analysis nodes to appear by the latter two methods means the analyst has the flexibility to take an action which does not appear on any menu.

The new node may be attached to any number of existing nodes, or to none at all. For example, the **PlotOfLogs ScatterPlot** was created by selecting the **ScatterPlot** template from the toolbox. The analyst was then prompted for the two required variables and asked to indicate where in the network the new node should be attached. The analyst chose to link it to the transformed data vectors on which it was based, but need not have. In point of fact, then, our model for a statistical analysis is a collection of disjoint acyclic directed networks.

3.0 Managing the analysis

Our model of a statistical analysis is a collection of acyclic directed networks, where every node in a network represents a step in the analysis. Each node is implemented as an object that is statistically meaningful and the visual interface provides a menu to indicate the actions usually taken from that step. The logical flow of the analysis is maintained by links between these objects.

Each object has two sets of pointers. One set, called AnalysisLinks, points forward to those objects that branch out of it in the analysis. The other set, the BackAnalysisLinks, points backwards to those objects that immediately precede it in the analysis. For example, **PlotOfLogs** in figure 1 has a single pointer in its AnalysisLinks (to the **BivariateLeastSquares**) and two pointers in its BackAnalysisLinks (to **LnBodyWts** and **LnBrainWts**). These links are typically established automatically.

To help manage the analysis, the analyst may rearrange these links at will. Dead ends can be pruned, nodes can be

rearranged to show more clearly the logic of some finding, and so on. There is also a facility to insert a **Memo** between any two nodes to record commentary - perhaps to highlight the reasons for pursuing a certain course. However, most analyses are more complex than the one presented here. Even with the ability to make and break links, it will not be long before the display gets out of hand.

There are two complementary needs. The first is to suppress some parts of the analysis from present consideration. If part of the network can be thought of as a self-contained unit, a sub-analysis say, then it should be possible to present it as such in the display, suppressing its detail from view. The second is to focus attention on just such a sub-analysis. For example, the analysis of figure 1 might possibly be a small analysis within a more general one that investigates many more attributes of mammals.

These management needs are addressed by **AnalysisMaps**. Figure 1 is an **AnalysisMap** named **Analysis of Mammals**. The relationship of an **AnalysisMap** to a statistical analysis is analogous to that of a roadmap to a system of roads. The map is not the system of roads, it is a picture of part of the system. Similarly, **AnalysisMaps** provide only a view of the network system that represents the analysis.

Like a roadmap, an **AnalysisMap** will not necessarily show all levels of detail. Figure 2 shows the **Analysis of Mammals** with some of its details suppressed from view.

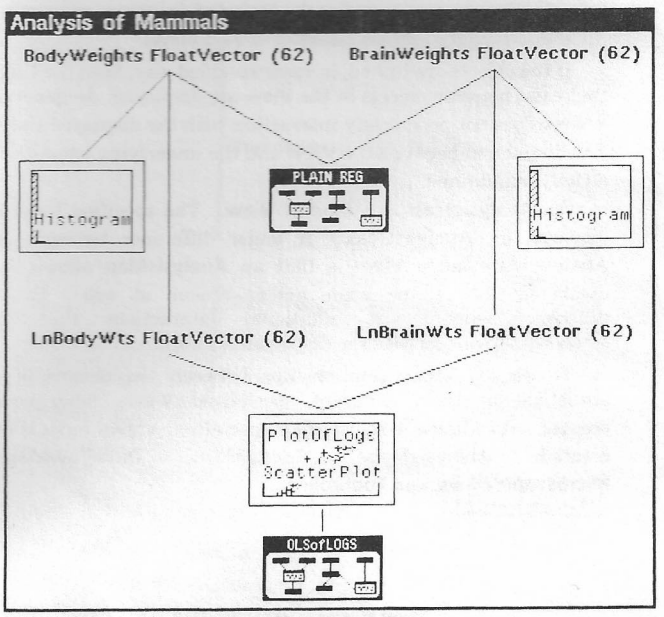

Figure 2. The **Analysis of Mammals** with some details suppressed

The two icons represent smaller **AnalysisMaps**, named **PLAIN REG** and **OLSofLOGS**, which contain more detail. These icons will produce menus in the same fashion as any other node in the display. *Zoom*ing in on them, however, will produce the displays of figure 3. These displays are identical in function to the display of figure 1.

The analysis can continue within either display of figure 3 without affecting the display of figure 2. This permits the analyst to focus attention on a sub-analysis like **OLSofLOGS** without complicating the bigger picture of **Analysis of Mammals**.

The suppression of detail, or alternatively the focussing of attention, can continue within **OLSofLOGS** or **PLAIN REG**. For example, all the residual plots of **OLSofLOGS** could be compressed into yet another **AnalysisMap**. The number of levels available is unrestricted.

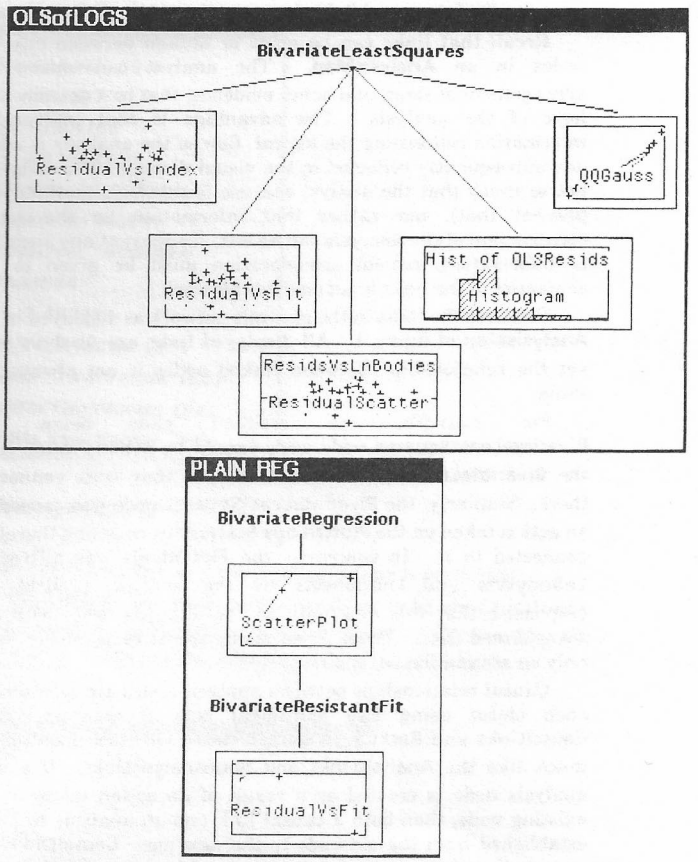

Figure 3. The **AnalysisMaps**: **OLSofLOGS** and **PLAIN REG**

The coverage of an **AnalysisMap** can be widened or narrowed as required. It is widened either by actions taken within the **AnalysisMap**, as indicated in Section 2, or by adding new analysis nodes. These new nodes may be newly constructed ones - by selecting a template from the toolbox or by evaluating a user-supplied function - or they may be existing nodes that are displayed in other **AnalysisMaps**. Two **AnalysisMaps** may in fact view common nodes. An **AnalysisMap** is prevented from including itself in its view (at any level of detail), but otherwise there is no restriction on its coverage. It can be narrowed only by explicitly removing (mouse-)selected nodes from its view. When **AnalysisMaps** are nested, a node removed from one will automatically appear in all those which envelop it.

AnalysisMaps allow the analyst to make and break links between nodes in its display. Note, however, that **AnalysisMaps** themselves do not maintain links to other objects. In figure 2, links to **PLAIN REG** are shown because AnalysisLinks exist from **BodyWeights** and **BrainWeights** to some nodes contained in **PLAIN REG**. Similarly, no links emanate from either **PLAIN REG** or **OLSofLOGS** because no nodes within either of them have AnalysisLinks to other nodes in the display.

This is an important point. **AnalysisMaps** are not part of the analysis network. The analysis network remains exactly as it was displayed in Figure 1. Like roadmaps, **AnalysisMaps** provide different views of the underlying network, they do not participate in it.

4.0 Link semantics

Recall that links can be made or broken between any two nodes in an **AnalysisMap**. The analyst can choose the arrangement of steps, and hence evidence, that best describes the logic of the analysis. The advantage is that interpretive information concerning the logical flow of the analysis is added and subsequently reflected in the visual display. The danger is not so much that the analyst can use faulty logic (nothing will prevent that), but rather that information on the actual construction of the analysis can be lost. To prevent any such loss of information, careful consideration must be given to the semantics of the links in an analysis network.

Consider the links in the analysis network as displayed in the **AnalysisMap** of figure 1. All displayed links are AnalysisLinks, yet the relationship between linked nodes is not always the same.

For example, the residual plots below the **BivariateLeastSquares** node were caused by actions taken from the **BivariateLeastSquares** node. Hence, they were connected there. Similarly, the **BivariateLeastSquares** node was caused by an action taken on the **PlotOfLogs ScatterPlot** node and therefore connected to it. In contrast, the **PlotOfLogs** was affixed to **LnBodyWts** and **LnBrainWts** by the analyst, probably to emphasise that this new path in the analysis deals with the transformed data. There is no strict causal relationship here, only an association.

Causal relationships between analysis nodes are recorded on each object using two additional sets of pointers called CausalLinks and BackCausalLinks. These links are maintained much like the AnalysisLinks and BackAnalysisLinks. If a new analysis node is created as a result of an action taken at an existing node, then both a causal link and an analysis link are established from the old node to the new one. CausalLinks are used to establish a causal relationship, while AnalysisLinks are used to show the logical flow of the analysis. Unlike AnalysisLinks, CausalLinks are permanent - they cannot be made or broken by the user. In this way, no amount of rearranging of AnalysisLinks will destroy information on causal relationships between analysis steps.

A simple association is said to exist between nodes if there exists an analysis link joining the two but no causal link. These links are established only by the analyst. Their function is to reflect the logical flow of the analysis, which is not necessarily a causal flow.

Some simple associations, like those AnalysisLinks from **LnBodyWts** and **LnBrainWts** to the **PlotOfLogs**, might be described more accurately as data flow links. However, to follow the pattern used with CausalLinks and establish an analysis link whenever a new analysis node uses existing data, would quickly complicate the display. For example, AnalysisLinks from **BodyWeights** and **BrainWeights** to each of **BivariateRegression**, **ScatterPlot**, and **BivariateResistantFit** would do more to obscure the logical flow of the analysis than to illuminate it. Since all objects have internal pointers to the data on which they depend, this information is not displayed as a link in the **AnalysisMap**. Instead, like many of the statistical results of an action (e.g. residual **FloatVectors** on a **BivariateResistantFit** or **BivariateLeastSquares** object), this detail is suppressed in the **AnalysisMap** and can be accessed through *Zooming*.

Thus, there are at least three possible meanings that might be given to an association between analysis steps. First, an analysis link asserts that one step follows logically from another in the analysis. These are managed entirely by the user. Second, a causal link shows direct causation between steps. These cannot be changed in any way by the analyst (either adding or deleting them). Since causation means that the second node must have followed directly from the first, an analysis link is also made. Finally, some associations strictly indicate the flow of data. Unlike causal links, this association does not automatically create an analysis link since it is not always relevant to the logic of the analysis.

5.0 Some interesting Views

The statistical analysis is carried out by interacting with, and manipulating, the contents of an **AnalysisMap**. Recall that **AnalysisMaps** are views of networks whose links are AnalysisLinks. Since other links exist between the steps of an analysis, it may be of interest to view the collection of networks as defined by each kind of link.

The generic counterpart of an **AnalysisMap** is something which we call a **View**. A **View** is just that, a view of a collection of objects. When *Zoomed*, it displays the objects in a mouse sensitive window that allows interaction both with the individual objects displayed and with the display as a whole.

Views can be *widened* to include any existing object (including other **Views**) provided such *widening* does not cause them to include themselves (either directly or indirectly). Alternatively, a **View** can be *narrowed* to exclude selected objects already in view. This latter option includes the possibility of having the selected objects placed in a new **View** inside the old one - this allows a **View** to compress some of its detail into a more refined **View**. In each **View**, the identity of all **Views** which immediately envelop it is saved. This allows any object excluded from a **View** to be immediately included in these enveloping **Views**, enforcing a scoping effect for nested **Views**.

If the objects are linked, in some specified way, then the links that exist between objects in the **View** are displayed. In general, a **View** does not permit any interaction with the displayed links. The distinction between the **View** and the underlying network is strictly maintained.

An **AnalysisMap** is a kind of **View**. The specified links it displays are AnalysisLinks. A major difference between an **AnalysisMap** and a **View** is that an **AnalysisMap** allows the displayed links to be made and/or broken at will Other differences amount to additional interactions that an **AnalysisMap** allows with the displayed network.

To display other relationships between the objects of a statistical analysis, different specialized **Views** have been created. As Figure 4 shows, five specialized **Views** have been created: **AnalysisMaps**, **CausalMaps**, **DataFlowMaps**, **MicroscopicViews**, and **ToolBoxes**.

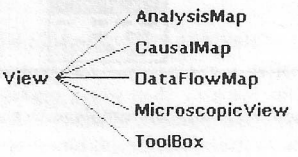

Figure 4. Specializations of **View**

As shown, each inherits all the behaviours of a **View** (*widening*, *narrowing*, etc.). In addition, each can have specialized behaviours of its own.

MicroscopicViews are the simplest **Views**: each represents a view of the internal structure of a single object. As such, they provide the finest level of detail accessible through *Zooming*. Figure 5 shows the **MicroscopicView** that results from *Zooming* in on the **BivariateLeastSquares** object in the example.

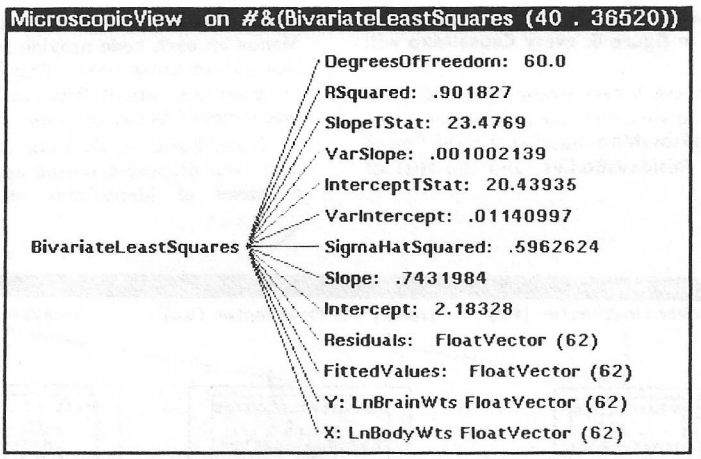

Figure 5. A **MicroscopicView** of the **BivariateLeastSquares**

As can be seen, the **BivariateLeastSquares** contains internal pointers to many other objects, including the X and Y vectors **LnBodyWts** and **LnBrainWts**, respectively. Each of these can again be *Zoom*ed in on, however, unlike other **Views**, no widening or narrowing of a **MicroscopicView** is permitted.

CausalMaps are **View**s whose specified links are CausalLinks. Figure 6 shows a **CausalMap** that contains all of the objects in the example analysis and the causal relations that link them.

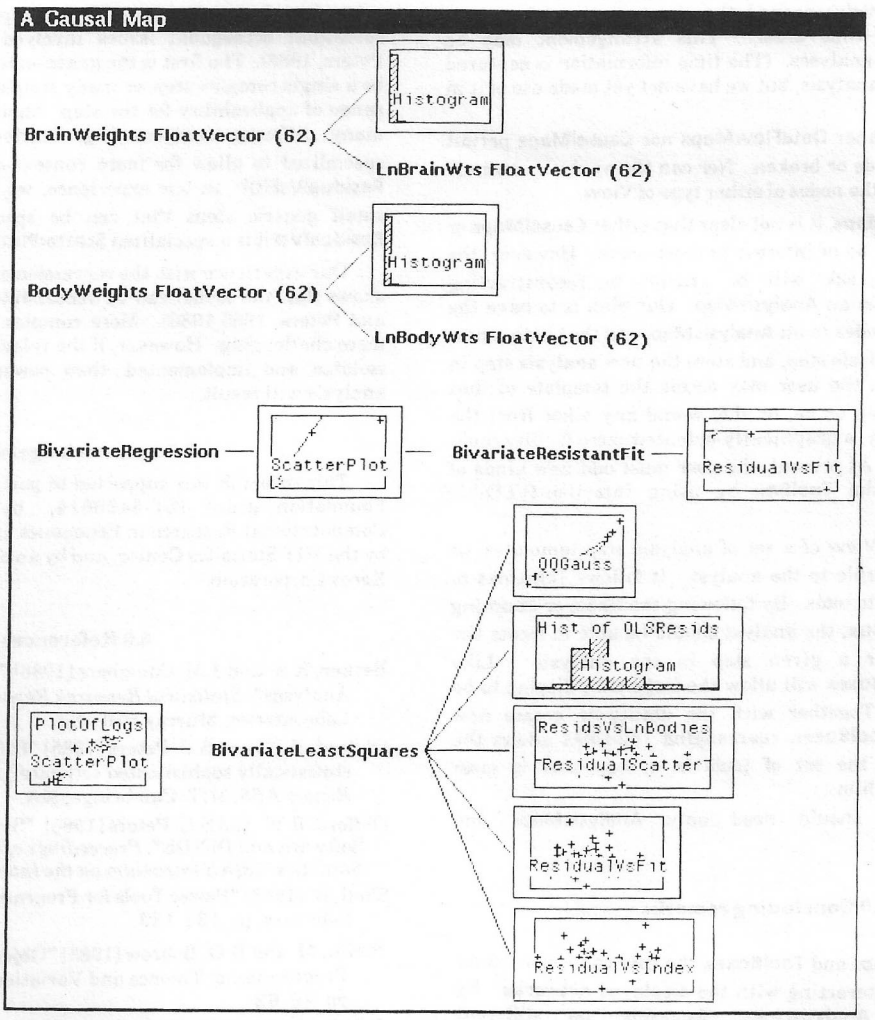

Figure 6. A **CausalMap** of objects in the mammalian analysis.

The direction of causation is from left to right. Here, four separate networks exist. As in figure 6, every **CausalMap** will contain only trees.

Similarly, **DataFlowMaps** are **Views** whose specified links indicate the identity of required variables for each object in the display. Figure 7 shows a **DataFlowMap** that displays the flow of data to the **PlotOfLogs**, the **ResidsVsBodies**, and the **Hist of OLSResids**.

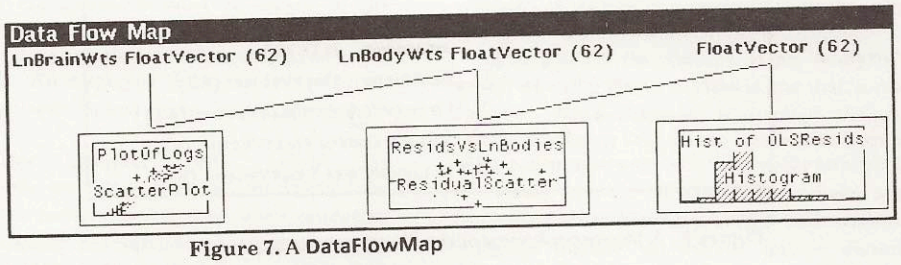

Figure 7. A **DataFlowMap**

The direction of flow is from top to bottom. The unnamed **FloatVector** in the top right of the display is the vector of residuals from the **BivariateLeastSquares** object. Becker and Chambers [1986] have suggested a similar graphic, except that they arrange the nodes around the circumference of a circle according to their time order. This arrangement may be preferred for larger analyses. (The time information is captured on each node in the analysis, but we have not yet made use of it in the display.)

Like **Views**, neither **DataFlowMaps** nor **CausalMaps** permit their links to be made or broken. Nor can the analysis continue by interacting with the nodes of either type of **View**.

Given **AnalysisMaps**, it is not clear that either **CausalMaps** or **DataFlowMaps** will be of interest to most users. However, the information they track will be crucial to reconstructing user-selected paths in an **AnalysisMap**. Our plan is to have the user select a set of nodes in an **AnalysisMap**, call that path a new kind of complex analysis step, and store the new analysis step in the **ToolBox**. Then, the user may access the template of that complex analysis step as he, or she, would any other from the **ToolBox**. In this way, a graphically-oriented macro facility could be made available. At present, the user must add new kinds of analysis steps to the **ToolBox** by using Interlisp-D/LOOPS programmes.

A **ToolBox** is a **View** of a set of analysis step templates, or tools, that are available to the analyst. It follows ToolLinks to organize the available tools. By following the links, or Zooming in on a nested **ToolBox**, the analyst should be able to locate the appropriate tool for a given step in the analysis. Like **AnalysisMaps**, **ToolBoxes** will allow the links they display to be broken or made. Together with the ability to create new **ToolBoxes** within **ToolBoxes**, rearranging ToolLinks allows the analyst to arrange the set of tools in a way that is most convenient to her, or him.

Most analyses should need only **AnalysisMaps** and **ToolBoxes**.

6.0 Concluding remarks

With **AnalysisMaps** and **ToolBoxes**, the user can construct an entire analysis by interacting with the displayed networks. By rearranging the AnalysisLinks, focussing on different **AnalysisMaps**, and adding commentary to individual nodes, the analyst can easily record the logic of the analysis as it progresses. Menus on each node provide immediate access to those actions likely to be taken next. This allows the analyst to concentrate more on the salient features of the analysis and less on the mechanics of its construction.

Note, however, that our model of statistical analysis, and hence our display-oriented approach, depends critically on the existence of identifiable steps in a statistical analysis. Identifying the appropriate steps for a given area of statistical analysis is crucial, and often quite challenging.

The principal difficulty lies in clearly defining the domain of each step. The data it needs access to, and the actions that can be taken from it, must be unambiguously prescribed. There are two somewhat orthogonal issues involved (see also Oldford and Peters, 1986). The first is the grain-size of the step. Should there be a single complex step, or many simple ones? The second is the range of applicability for the step. Should the step be generic to many different analyses (e.g. **ScatterPlot**), or should it be specialized to allow for more context-dependent actions (e.g. **ResidualVsFit**)? In our experience, we have preferred to select small generic steps that can be specialized as needed (e.g. **ResidualVsFit** is a specialized **ScatterPlot**).

Our experience with the regression of one variable on another shows that the model can be successfully implemented (Oldford and Peters, 1985,1986). More complex statistical areas will be more challenging. However, if the relevant analysis steps can be isolated and implemented, then powerful new tools for data analysis will result.

7.0 Acknowledgements

This research was supported in part by the National Science Foundation grant IST-8420614, by the MIT Center for Computational Research in Economics and Management Science, by the MIT Statisitcs Center, and by an equipment grant from the Xerox Corporation.

8.0 References

Becker, R.A. and J.M. Chambers [1986] "Auditing Data Analyses", *Statistical Research Report No. 25*, AT&T Bell Laboratories, Murray Hill , NJ.

Oldford, R.W. and S.C. Peters [1985] "DINDE: Towards more statistically sophisticated software", *CCREMS Technical Report # 55*, MIT, Cambridge, MA.

Oldford, R.W. and S.C. Peters [1986] "Statistically Sophisticated Software and DINDE", *Proceedings of Computer Science & Statistics: 18th Symposium on the Interface.*

Sheil, B. [1983] "Power Tools for Programmers", *Datamation*, February, pp. 131-143.

Stefik, M. and D.G. Bobrow [1985] "Object-Oriented Programming: Themes and Variations", *The AI Magazine, 5*, pp. 40-62.

Teitelman, W. and L.M. Masinter [1981] "The Interlisp programming environment", *IEEE Computer, 14*, pp. 25-34.