STATISTICALLY SOPHISTICATED SOFTWARE AND DINDE

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

Abstract

We describe a prototype system, which we call DINDE, and the directed network model of statistical analysis on which it is currently based. DINDE is a highly interactive display oriented system where the user carries out the analysis by building and maintaining a network representation of it. An example analysis is used to describe this interaction and the analysis management tools required.

1.0 Introduction

By statistically sophisticated software, we do not mean software that implements a sophisticated statistical method, but rather, software that contains information on how and when that method is most frequently used in practice.

As shown in Figure 1, there are at least three

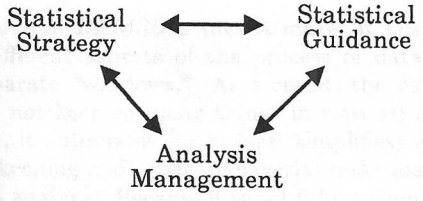


Figure 1: Three interrelated objectives

reasons why one might attempt implementation of such software: first, to have software which guides the user to a better statistical analysis (e.g. Gale and Pregibon[1982], Oldford and Peters[1984]); second, to use the software as a medium for studying statistical strategy (e.g. Pregibon[1985] or Oldford and Peters[1984,1985ab]); and, third, to have software which helps the user manage the analysis (e.g. Carr et al [1984], Becker and Chambers[1985], Oldford and Peters[1985b]).

These objectives are interrelated and software for one often leads naturally to software for another. For example, to successfully guide the analysis one needs to understand and implement the supporting statistical strategies. To study the strategies of good statistical data analysis with software, one needs to be able to manage the analysis. In developing DINDE, we have repeatedly found ourselves concentrating on each objective in turn; advancement toward one objective has often produced insight on one of the others.

Whether the chief interest is to *guide*, *study*, or *manage* the analysis, some model of an analysis is required. The next two sections address this question rather generally. The remaining sections describe the current implementation of one such model in DINDE. Like any model, the current model of statistical data analysis used in DINDE is temporary and will improve with experience. The last section indicates some of the modifications we already foresee.

2.0 What's an Analysis?

The simplest, and least satisfying, view of a statistical analysis is as a *specified sequence* of steps. An example would be regression modelling by a forward selection procedure, where variables are added to the current regression model one at a time according to some criterion.

This model of statistical analysis seems to underlie the batch-oriented statistical packages of the late 1960s and early 1970s, or, at least, to underlie their common usage. The sequence of steps to be taken in the analysis is defined in advance and the corresponding set of packaged routines is run. In light of the results, this sequence may be modified and the resulting new set of procedures run. Each run is regarded as a different analysis. The refinement of the analysis continues until the analyst is satisfied that the final one is "correct" for the problem. In this paradigm, novices typically would not substantially modify their original analysis.

A more accurate view of statistical analysis, based largely upon a scientific modelling paradigm as expressed, for example, by Box[1976], is that it is an iterative procedure whereby statistical models are alternatively fitted and criticised.

Unrolling this iterative loop produces a different kind of sequential process, one whose steps are not predetermined. Instead, the analyst decides what to do next based upon the results of the preceding step. Rather than a specified sequence that is continually refined, this model of an analysis is a *dynamic sequence*, one that grows as more is learned about the problem and the data. The analysis is represented by the entire sequence, not just by a final revision. Here, novices would typically produce shorter sequences than would experts.

As such, this model fits well with modern interactive statistical systems like S (Becker and Chambers [1984]), where each step of the analysis corresponds to a command issued to the system. After examining the results of each command, the analysis is grown by issuing another command. A macro facility is usually available to allow the user to compress many small sequential steps in the analysis into a single larger one that is easier to comprehend, and use, as a unit. In this way, new analysis steps are defined by the analyst. Diaries reinforce this model of analysis by recording the entire sequence of steps.

The dynamic-sequence model suggests that results and information from actions taken early in the analysis influence later actions only through the chain of steps given by the time ordering. But this is not an accurate description of an analysis. At any step, a number of different actions can be, and often are, taken. A model of analysis based only on the time order of actions hides this logical relationship between actions, and, hence, is seriously incomplete.

This is an important shift in focus: from the time ordering of statistical and arithmetic procedures to the conceptual steps of the analysis and the logical

connections between the steps. Now, the simplest model of the analysis is a *tree*, where branches indicate a logical connection between one step and a number of others. For example, at different times, different actions or decisions may be taken from one step, resulting in many branches from it. With this model, novice analysts would likely produce short, sparse trees and expert analysts long, bushy trees.

A little reflection, however, shows that the tree model also falls short. Suppose that two branches of the tree represent two different sub-analyses that are pursued in parallel. It may happen that a new tack is taken in the analysis that is based on the combined results of both independent sub-analyses. Where should this new sub-analysis be attached? The obvious answer is to attach it to both of the previous ones, forcing the whole analysis to become a *directed network* rather than a tree.

This directed network model of statistical data analysis is currently the basis for DINDE. While it provides a better description than any of the previous models, it too has shortcomings. We discuss some of these in the last section, and indicate our planned modifications to the network model.

3.0 What are the steps again?

Implicit in the models considered above is the assumption that every analysis has identifiable steps: decision points where some action is taken. Further, it is assumed that many of these steps are generic enough to be usefully recorded. What, then, are the steps?

With current statistical systems, the steps are equivalent to the commands that are issued to the system. The first thing to notice is that the steps have varying granularity. The smallest grains include those steps where the actions are simple, arithmetic ones taken on scalars, vectors, matrices, and the like. Good statistical systems will always allow actions to be taken at this low level of analysis. Larger grains include strictly statistical actions, like regressing y on x, where the lower level steps needed to accomplish the task are suppressed from consideration. The regression step is really an abstraction of many lower-level analysis steps, an abstraction that becomes a powerful tool for the analyst.

More abstract steps are typically more powerful (i.e. do more for the analyst), but also have more restricted ranges of application (e.g. regression is a powerful tool but has smaller range of application than the matrix operations used to construct it). In designing useful steps for statistical analysis, there is always a potential tension between the range and the power of a newly proposed step.

In DINDE, our working philosophy has been to begin with steps that are reasonably generic and to increase their power in two ways that do not restrict their range of application.

First, we specialize some steps to become more context specific. The specialized steps are to be used in place of more general steps when the context warrants it. Therefore, the necessarily smaller range of applicability of the specialized steps does not inhibit the analysis. Instead, given the right context, they become powerful tools.

Second, in each step, information is incorporated as to which steps are often taken next. This simple addition makes the step more powerful at no loss to its range of application.

In DINDE, commands sometimes produce steps, but, steps are never commands. The steps in DINDE are quite different; they are collection points for possible actions (commands). Instead of specifying a sequence of actions to be taken, the abstraction in DINDE is to collect together a set of possible actions and the information that may help the analyst decide among them. A sequence of actions identified and captured in DINDE would simply be a more abstract action, not a step.

At present, the steps that have been developed in DINDE are either analysis goals or analysis artifacts.

For example, an analysis goal might be a reasonable description of the regression of y on x (i.e. the conditional expectation of y given x). This goal is represented in DINDE by the step *BivariateRegression* (bivariate since two variables are involved). A variety of information is immediately available at this step: which vectors x and y refer to, what set of actions are generally reasonable to take next (various plots and fitting routines), and which of these actions it is considered wise to take first (visually inspect the data via a scatterplot and histograms).

Choosing to do the scatterplot of y versus x would produce a *Scatterplot* as an artifact. Again, since it is a new step, particular actions would be made available for this kind of artifact (like fitting a straight line or a smooth curve to the points in the plot).

Clearly, a number of steps are necessary for an analysis with even the relatively simple goal of describing the regression of one variable on another. The challenge, then, is to make the steps generic enough to be useful in a variety of analyses.

4.0 DINDE

The challenge of DINDE is to produce and implement a model for statistical analysis that is both reasonably accurate and natural to use. Sections 2 and 3 indicate the underlying model; here, and in the sections which follow, we discuss its implementation and use.

DINDE is an enrichment of an extensive interactive programming environment: Interlisp-D with LOOPS which runs on the Xerox 1108 personal workstation (see Teitelman and Masinter [1981], Stefik et al [1983]). 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 designing and building DINDE.

The hardware is described in more detail in Oldford and Peters [1985b]. We note only two features here: first, an illusion of an infinitely large memory is maintained for the user (1.5 - 3.5 MBytes of real memory, 32 MBytes virtual), and second, the display is a high resolution bit-mapped display (about 1000 by 1000 individually addressable pixels) that can be interacted with using a "mouse" pointing device.

The software environment is at least as important as the hardware. Interactive programming environments are the most appropriate and productive locales for doing the sort of experimental programming that is involved in building a system like DINDE (and, we would argue, in carrying out statistical data analysis). This point is forcefully argued by Sheil [1983]. We have found the object-oriented programming paradigm, as available in LOOPS, to be especially useful: it is the backbone of DINDE (see Goldberg and Robson [1983], Bobrow and Stefik [1983], or Stefik and Bobrow [1985] for details on this programming paradigm).

In the object-oriented paradigm, there are classes, which contain generic properties and behaviours for a large "class" of individual objects, and objects, which are individual "instances" of a particular class. For example, the class *Car* would represent the common properties and behaviours of all *Cars*, while the object *MyCar* would be a particular instance of the generic class *Car*, as would *EdsCar*, *KarensCar*, and so on. In DINDE, each analysis step (goal or artifact) is represented as a class; the steps actually taken in a particular analysis are objects, instances of their corresponding classes.

The idea in DINDE, then, is to select the kind of step (class) which one wants to take next and to incorporate an instance of it (representing the step actually taken) at an appropriate place in the analysis (directed network).

The set of possible steps (i.e. classes) that can be taken in an analysis are displayed in an interactive window called the *toolbox*. Using the mouse, the analyst selects a step from the toolbox, as necessary, and attaches it (now an object) to the appropriate place in the analysis.

Recall that our model of an analysis is a directed network whose nodes are the steps actually taken. The analyst works with this model within another window called an *analysis map*. Here, the network representing the analysis is actually displayed and the analysis progresses by interacting with the network via the mouse.

This mouse interaction, called "mousing" in what follows, is possible in both the toolbox and the analysis map. In both windows the mouse behaviours follow some general principles. Figure 2

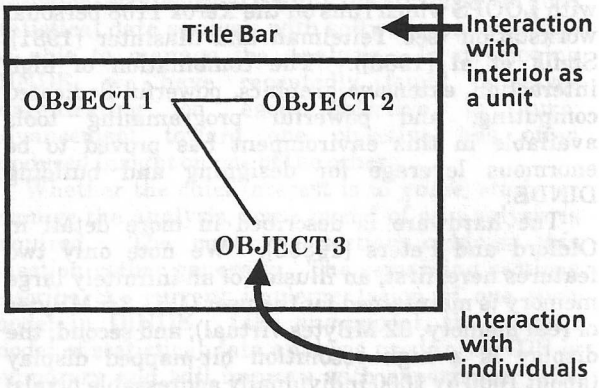


Figure 2: Mouse sensitive areas of DINDE windows

shows a generic window in DINDE and the basic mousing that can be done with them. There are two mouse sensitive areas in these windows: the title bar and the body. Mousing in the title bar allows interaction with the displayed contents as a whole; mousing on individual objects displayed in the body allows interaction with the mouse-selected object.

The mouse that is used on the Xerox 1108 has two buttons, yielding three different combinations of depressed buttons: left, right, and middle (both buttons) depressed. Depressing any one of these typically causes a menu to pop up (at the mouse position) from which one of a number of items can be selected. Selecting an item causes some action to be taken.

We try to follow the principle that a button combination should yield a similar kind of menu regardless of what is being moused. So, in the toolbox and the analysis map, left-buttoning always produces menus whose items have something to do with either accessing or storing information on the thing being moused. If the title bar is moused, then the information pertains to the toolbox or analysis map itself. If an individual object in the body is moused, then the information pertains only to that particular object. Right-buttoning always brings up a menu containing items that allow the user to manipulate the window (move it, reshape it, shrink it, etc.). Middle-buttoning causes menus to appear whose items indicate the messages that the thing selected can respond to. Typically, these will be action items such as "fit a smooth curve to your points" if the thing selected is a *Scatterplot* object.

The toolbox and the analysis map are discussed in turn in the next two sections.

5.0 The Toolbox

All of the classes of objects that can be used in a statistical analysis in DINDE are arranged in the toolbox and displayed to the user. Figure 3 shows the contents of the toolbox as it currently exists in DINDE.

We have tentatively established a coarse partition of the possible classes into five basic element types: (i) *Data* (currently represented as *Arrays* or *TreeStructures*), (ii) *Graphics*, (iii) *Situations*, (iv) *Models* (e.g. probability models), and (v) *Tables*. To date, only the first three of these exist in DINDE. Only these were necessary to build the prototype regression analysis, but we anticipate that eventually *Model* and *Table* representations will also be required.

In the toolbox of Figure 3, the classes are displayed as nodes on several trees. Traversing the trees from left to right is equivalent to moving from generic steps to more specialized ones. For example, *BooleanArrays*, *StringArrays*, and *FloatArrays*, are all specialized *Arrays* (specialized to have array elements whose values must be booleans, strings, and floating point numbers, respectively). Once more tools are available in DINDE, other arrangements may be of interest (different indexing depending on interest). Indeed, it will likely become desirable to group tools together into smaller toolboxes, or toolkits, as the number of tools grows. At present, however, the classes are displayed only

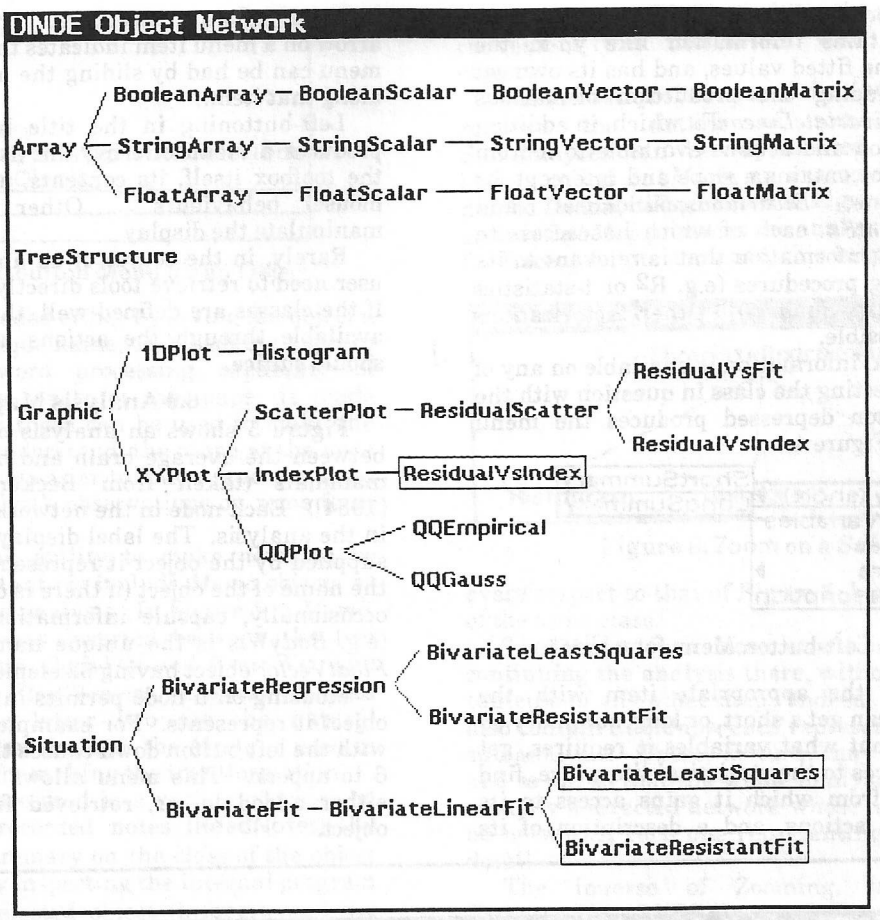


Figure 3: The DINDE Toolbox

according to their specialization.

In general, a specialization has access to all the information and actions available to a more general step, and more. Thus, what seems to be a counterintuitive relationship between a *FloatMatrix* and a *FloatVector* makes sense, because a *FloatMatrix* has access to actions, such as taking the singular value decomposition of itself, which would make little sense for a *FloatVector*.

Those specializations which have been undertaken to increase the power of steps are more intuitive. For example, *ResidualScatter* is a specialization of *Scatterplot* which always has residuals plotted along the vertical axis. Given this context, *ResidualScatter* has access to actions which would not make sense for an arbitrary *Scatterplot*, like the ability to smooth the positive and negative values of the vertical coordinates (although smoothing all the vertical coordinates will be available to both kinds of plots). Similarly, the ResidualVsFit plot has access to actions which are helpful in determining whether the residual error is heteroscedastic.

Notice that these relations between the classes do not really conform to trees. LOOPS allows one to create classes that are specializations of more than one class (e.g. *ResidualVsIndex*). Such classes are identified by having a box drawn around them whenever they are repeated in the display. They have access to all information and actions that are available to any of their "parent" classes.

This ability to mix together classes encourages the abstraction of common aspects of different kinds of analysis steps. The abstractions are then represented as generic classes that can be usefully "mixed into" more than one step. This is perhaps one of the most challenging aspects of creating a sophisticated system like DINDE. It requires an identification and grouping of the elements that are practically important in a statistical analysis.

The nature of this research can be seen by considering those analysis steps that are classified as *Situations*. There are five different classes, of which only one can really be regarded as a goal (*BivariateRegression*), the others are better described as artifacts.

The analysis step *BivariateRegression* represents the goal of regressing y on x. It contains the necessary information on which vector is y and which is x, has access to various plotting methods (*Scatterplot* and *Histograms* of y and x), various fitting procedures (a straight line via least-squares or an outlier resistant procedure and a running linear least-squares smooth). Selecting any of these actions will produce an artifact (a plot or a fit), and hence a new analysis step. There are four kinds of fit artifacts. In order of increasing specialization they are as follows: *BivariateFit*, which is used to

represent arbitrary fits (such as those from smoothers), contains information like y, x, the residuals, and the fitted values, and has its own set of actions, including the production of various residual plots; *BivariateLinearFit* which, in addition to the information and actions available to it from *BivariateFit*, also contains a slope and intercept for the fitted line; *BivariateLeastSquares* and *BivariateResistantFit*, each of which has access to more specialized information that is relevant to its particular fitting procedures (e.g. R^2 or t-statistics for *BivariateLeast- Squares*). Other factorizations are certainly possible.

In the toolbox, information is available on any of these steps. Selecting the class in question with the left mouse button depressed produces the menu shown below in Figure 4.

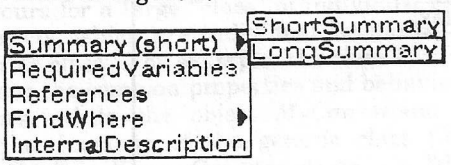


Figure 4: Left-button Menu for a Class

By selecting the appropriate item with the mouse, the user can get a short, or long, summary on that class, find out what variables it requires, get relevant references to the statistical literature, find out the classes from which it gains access to its information and actions, and a description of its internal software structure. (Note that a right arrow on a menu item indicates that a more detailed menu can be had by sliding the mouse to the right, along that item.)

Left-buttoning in the title bar of the toolbox produces a menu offering the user information on the toolbox itself, its contents, and the associated mouse behaviours. Other mouse buttons manipulate the display.

Rarely, in the course of an analysis, should the user need to retrieve tools directly from the toolbox. If the classes are defined well, then the tools made available through the actions at any given step should suffice.

6.0 Analysis Maps

Figure 5 shows an analysis of the relationship between the average brain and body weights of 62 mammals (taken from Becker and Chambers [1984]). Each node in the network represents a step in the analysis. The label displayed at each node is supplied by the object it represents, and consists of the name of the object (if there is one), its class, and, occasionally, capsule information on its contents (e.g. *BodyWts* is the unique name of a particular *FloatVector* object having 62 elements).

Mousing on a node permits interaction with the object it represents. For example, selecting a node with the left button down causes the menu of Figure 6 to appear. This menu allows information to be either added to, or, retrieved from, the selected object.

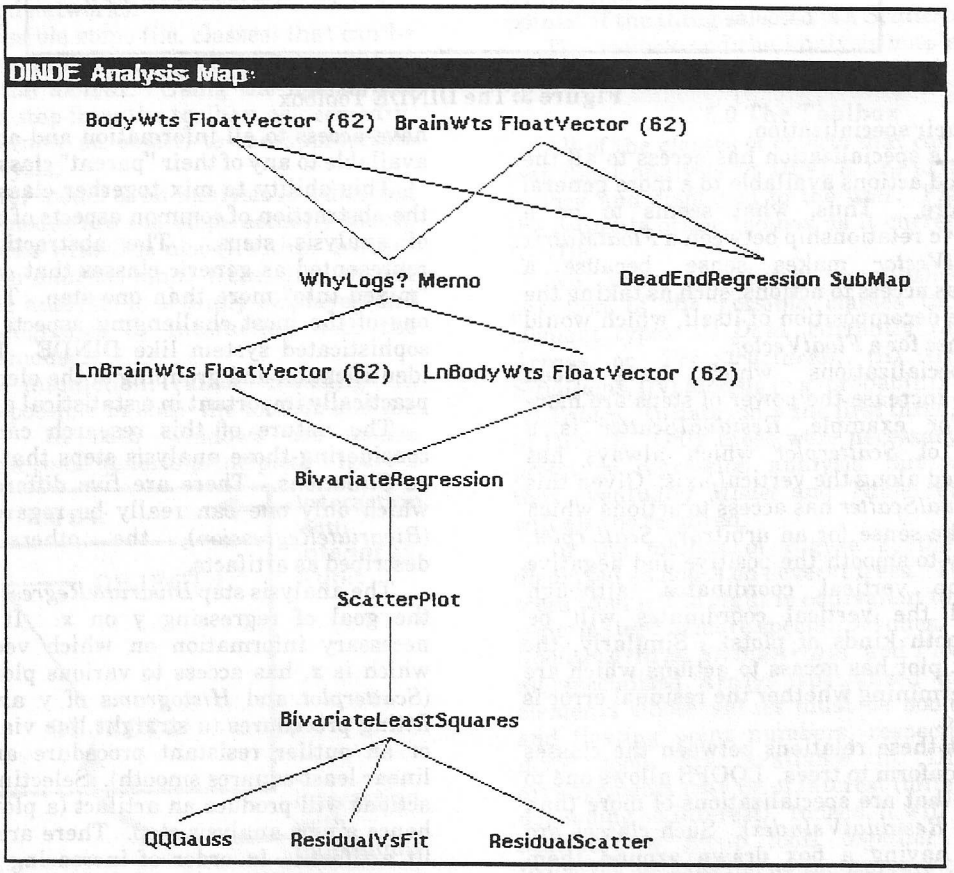


Figure 5: An Analysis Map

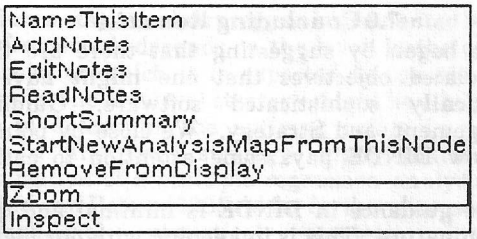


Figure 6: Left-button Menu for an Object

Information is added either by giving the object a meaningful and unique name, or, by adding notes to the object (the word processing capability of Interlisp-D used to construct this paper is made available). Both of these can be used to make the analysis easier to understand: the name at the node can make the display easier to follow, and the notes can record the analyst's observations on some facet of the analysis.

We consider the ability to make notes to be important enough that we include *Memo* objects as possible steps in the analysis. In Figure 5, a *Memo*, called *WhyLogs?*, was inserted between the two original data vectors, *BodyWts* and *BrainWts*, and the two derived *FloatVectors*, *LnBodyWts* and *LnBrainWts*. The latter two are the natural logarithms of the raw data, so the *Memo* is used to record the reasons for making the transformation.

Information is accessed in a variety of ways: by reading the user-recorded notes (ReadNotes), by printing a short summary on the class of the object (ShortSummary), by inspecting the internal program structure of the selected object (Inspect), and, by examining the detail contained in that node (Zoom).

The last of these is uniformly used in DINDE to access further detail on any node in the analysis. "Zooming" on a *Graphic* (e.g. *Scatterplot* or *QQGauss*) will cause a window containing the plot to appear. Zooming on other objects produces a window containing the "Zoomed" object and the data it can access. Figure 7 shows the effect of Zoom on the *BivariateLeastSquares* node of the map in Figure 5. (As in other DINDE windows, the mouse provides convenient way interaction with the displayed objects.)

The most important use of the Zoom facility is to retrieve the details of some sub-analysis. Sub-analyses can be represented in DINDE as objects called *SubMaps*, an example being the one named *DeadEndRegression* in Figure 5. Zooming in on this *SubMap* produces the analysis map of Figure 8. Except for its contents, this map is identical in every respect to that of Figure 5; both are instances of the same class.

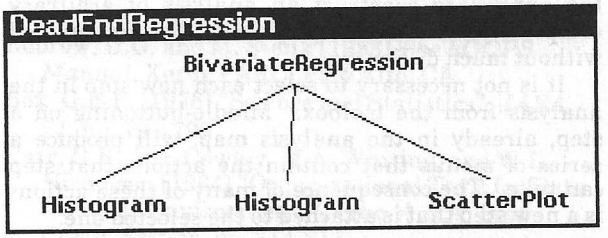


Figure 8: Zoom on a *SubMap*

The analyst can focus attention on this map, continuing the analysis there, without affecting the contents of any other map. Indeed, this map might also contain *SubMaps*, each representing a yet finer sub-analysis. These, in turn, may contain others, and so on, so that the whole analysis, in DINDE, is actually a directed network whose nodes might also be networks, each one representing a new level of detail.

The inverse of Zooming, in DINDE, is Compression. Middle-buttoning on the title bar of any analysis map produces a menu whose items correspond to operations on the displayed network. One of these items is Compress. Once selected, the user is required to identify nodes in the analysis (usually by mouse-selecting them) to be compressed into a single *SubMap*. All of the relationships between these nodes are maintained, so that Zooming on the new *SubMap* will reproduce the necessary detail.

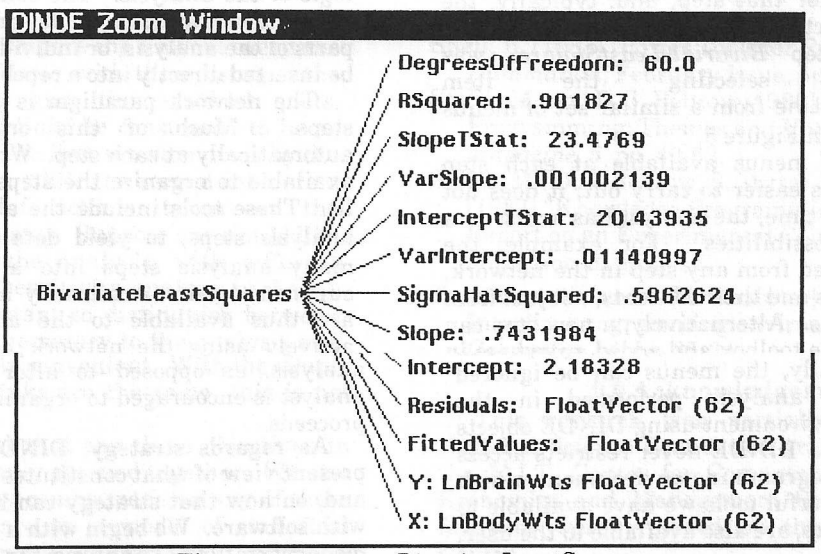


Figure 7: Zoom on *BivariateLeastSquares*

The other middle-button menu items from the title bar include the following: AddAnalysisNode which allows the user to select a new step from the toolbox and attach it anywhere in the network, MakeLink and BreakLink wich allow the user to make and break links in the network in order to make the analysis easier to understand, and, InterposeMemo, which allows the user to insert a memo between two nodes in the network. Together with Compress, these network tools should enable the analyst to construct an analysis of arbitrary complexity, whose display can be understood without much difficulty.

It is not necessary to select each new step in the analysis from the toolbox. Middle-buttoning on a step, already in the analysis map, will produce a series of menus that contain the actions that step can take. The consequence of many of these actions is a new step that is attached to the selected one.

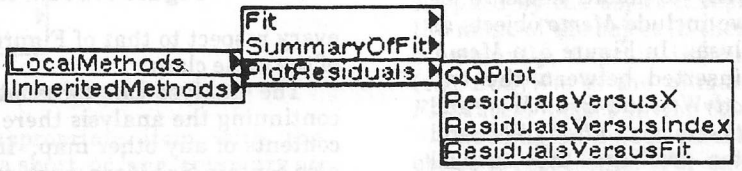


Figure 9: Some Menus from *BivariateLeastSquares*

Figure 9 shows some of the menus of actions that are accessible from a *BivariateLeastSquares* step (as in Figure 5). Here, the mouse has been moved to the right over the item LocalMethods, in the first menu, and over the item PlotResiduals, of the second menu, to display a menu of the possible residual plots that can be performed. The three steps *QQGauss*, *ResidualVsFit*, and *ResidualScatter*, of Figure 5, were produced by selecting from this menu the items QQPlot, ResidualsVersusFit, and ResidualsVersusX, respectively.

All steps in the analysis have access to menus that make sense for that step, and, typically, the analysis is constructed by selecting items from these menus. The step *BivariateLeastSquares* was produced by selecting the item AddALeastSquaresLine from a similar set of menus on the Scatterplot in Figure 5.

The system of menus available at each step makes the analysis easier to carry out; it does not restrict it. At any time, the analyst has access to a wide range of possibilities. For example, the analysis can proceed from any step in the network, including *SubMaps* and their contents, not just from the most recent one. Alternatively, a new step can be selected from the toolbox and added anywhere in the network. Finally, the menus can be ignored entirely and the analysis performed in the Interlisp/LOOPS environment using DINDE objects as they seem useful. DINDE never restricts access to the underlying programming environment, which means that the powerful tools we have available to us, as system builders, are also available to the user, as an analyst.

7.0 Concluding Remarks

We began by suggesting that there are three interrelated objectives that one might have for statistically sophisticated software: Guidance, Management, and Strategy. We close by pointing out how DINDE pays some attention to each of these.

The guidance in DINDE is minimal and quite local in nature. This is in keeping with our view as to what sort of guidance it is possible to competently give in a statistical analysis (see Oldford and Peters [1985a] for further discussion).

The guidance consists of the identification of the useful steps in an analysis, made available in the toolbox, and, of the actions and suggestions made available via menus at each analysis step. At present, the guidance is data-independent, in the sense that it does not depend on the data in hand. This does not rule out the possibility of data-dependent guidance, provided it too is of a quite local nature (e.g. noticing collinearity in regression).

Management of the data analysis is made easier in DINDE by having the analyst work with identifiable steps within a network metaphor for a statistical analysis.

With identifiable analysis steps, menus can be used to make those actions, which are often taken next, immediately available to the analyst. Further, notes can be added at each step, or inserted between steps, which will help the analyst recall the logic of the analysis. The computing environment also allows the notes, plots, and even snapshots of parts of the analysis, or individual analysis steps, to be inserted directly into a report.

The network paradigm is used to organize the steps. Much of this organization is done automatically at each step. When it is not, tools are available to organize the steps as the analyst sees fit. These tools include the ability to Zoom in on analysis steps, to yield detail, and, to Compress many analysis steps into a single *SubMap*, to suppress detail. Arbitrarily many levels of detail are thus available to the analyst. Finally, by actively using the network paradigm during the analysis, as opposed to after the analysis, the analyst is encouraged to organize the analysis as it proceeds.

As regards strategy, DINDE is based on our present view of what constitutes statistical strategy, and, on how that strategy can be fruitfully studied with software. We begin with a simple, yet general, model of statistical analysis: the directed network.

How a statistical strategy is implemented within this framework depends on the level, in the overall analysis, at which it is expected to operate. For example, one low-level strategy might be a heuristic used to determine the outlying points in a plot, while a high-level strategy might address the organization of a multiple regression analysis (see Oldford and Peters [1985ab] for further discussion).

High-level statistical strategy involves specifying the basic elements of the network model: the nodes, or analysis steps, and, the links which join them. In DINDE, the steps are represented as objects, and each object contains a specified set of actions that can be taken at that step. Thus, the analyst is strategically encouraged to link the present step to steps which result from taking the offered actions. Even stronger encouragement is provided through Suggestions, an action which produces canned text on what it is often considered wise to do first at this step. A low-level strategy is more likely to be implemented as an available action at some step.

Of the three interrelated objectives, that of using software to study the strategies of practical statistical analysis has been our primary focus. It is hoped that software, like DINDE, might provide a useful tableau on which statistical strategies can be recorded, and hence studied. To this end, we see that the model used in DINDE can be improved in two ways.

First, the fundamental data types in statistical practice are not vectors or matrices; these are artifacts of the mathematical analysis. More statistically meaningful are records on individuals, batches of numbers associated with a variate, and the dataset formed by combining many records or many batches. We are currently working on implementing such statistical objects as the fundamental *Data* objects in DINDE. (This in no way precludes the use of matrices and vectors to carry out the arithmetic.)

Second, the directed network model of analysis is not quite rich enough to suit our purposes. The difficulty is that not all links have the same meaning. For example, consider again the analysis map of Figure 5, and compare the links between the *FloatVector*s and the *Memo*, with those between the *BivariateLeastSquares* and the residual plots. Should the first set of links be considered to be as strong as the second? The first set were made by the user inserting a *Memo*, while the second set were a direct consequence of actions taken at the *BivariateLeastSquares* step. Suppose one would like to repeat a path in the analysis, with different values for the data (a sensitivity analysis perhaps), then it becomes important to distinguish between those links which are necessary to the analysis and those which are merely convenient. With the simple network, where all links are the same, this is not possible.

In closing, we feel that the three objectives in Figure 1 can be fruitfully pursued together. We also feel that real headway can be made on any of these by basing the software on a model of statistical analysis that is reasonably accurate, and, natural to use. DINDE is one such attempt that is currently based on a simple network paradigm for statistical analysis. As we improve the underlying model, DINDE will come closer to meeting these objectives.

8.0 References

Becker, R.A. and J.M. Chambers [1984] *S An Interactive Environment for Data Analysis and Graphics*, Wadsworth, Inc., Belmont CA.

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

Bobrow, D.G. and M. Stefik [1983] *The LOOPS Manual*, Xerox PARC: Palo Alto, CA.

Box, G.E.P. [1976] "Science and Statistics", *JASA*, *71*, pp. 791-799.

Carr, D.B., P.J. Cowley, M.A. Whiting, and W.L. Nicholson [1984] "Organizational Tools for Data Analysis Environments", *Proc. A.S.A. Stat. Comp. Section*, pp. 214-218.

Gale, W.A. and D. Pregibon [1982] "An Expert System for Regression Analysis", *Proc. 14th Symposium on the Interface*, pp. 110-117.

Goldberg, A. and D. Robson [1983] *Smalltalk-80: The Language and its Implementation*, Addison-Wesley: Reading, MA.

Oldford, R.W. and S.C. Peters [1984] "Building a Statistical Knowledge-based System with Mini-MYCIN", *Proc. A.S.A. Stat. Comp. Section*, pp. 85-90.

Oldford, R.W. and S.C. Peters [1985a] "Implementation and Study of Statistical Strategy", to appear in *Artificial Intelligence in Statistics - Statistics in Artificial Intelligence* (W.A. Gale, editor), Addison-Wesley, Reading MA.

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

Pregibon, D. [1985] "A DIY Guide to Statistical Strategy", to appear in *Artificial Intelligence in Statistics - Statistics in Artificial Intelligence* (W.A. Gale, editor), Addison-Wesley, Reading MA.

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

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

Stefik, M., D.G. Bobrow, S. Mittal, and L. Conway [1983] "Knowledge Programming in LOOPS: Report on an Experimental Course", *The AI Magazine*, *3*, pp. 3-13.

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

9.0 Acknowledgements

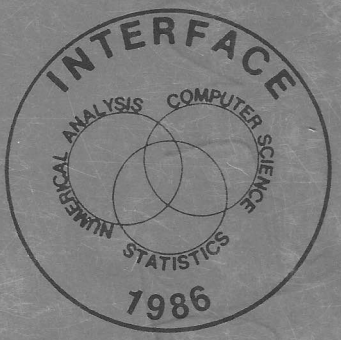
This research was partially funded by the National Science Foundation grant IST-8420614 to the M.I.T. Center for Computational Research in Economics and Management Science. Computer facilities were provided by the M.I.T. Statistics Center.

Computer Science and Statistics:

Proceedings of the 18th Symposium on the Interface

March 19–21, 1986

Thomas J. Boardman
editor



American Statistical Association • Washington, D.C.