Computational thinking for statisticians: Training by implementing statistical strategy

R.W. Oldford
Department of Statistics & Actuarial Science
University of Waterloo

Abstract

Students of statistics should be challenged to discover the possibilities computational technology has to offer empirical investigation. This paper describes recent experience with a course in statistical computing that tries to do just that.

Features of the course and the computing environment which would allow it to be replicated elsewhere are described. In particular, group projects which have the students design and develop software that implements a largish statistical strategy seem promising. Such a project forces the students to synthesize their statistical knowledge into a workable strategy. In so doing, they are exposed to a wide variety of topics, computational and statistical, at various levels of detail. Actually producing a software system challenges the computational creativity of the student, whatever their level of computational expertise.

The course experience, the base software environment, the statistical strategy developed by the students, and their software implementation are described.

1 The Problem

The computer provides a medium to be continually explored. Computer Science recognizes this and consequently occasionally wanders into areas of statistical interest (e.g. uncertain reasoning in expert systems, neural nets, data mining). Statistics follows, critically assesses, and sometimes improves the methodology. While assessment and improvement are important, it would seem that responsible leadership in empirical methods would benefit from expertise in both statistics and computer science. The need to train computer science students in statistical thinking is obvious to statisticians but somehow the need to train statistics students to think like computer scientists (possibly the more promising path) is rarely entertained.

Consider the training received in a good undergraduate statistics programme. By the end of it, students have some appreciation of the role of the computer in statistical analysis — it provides routine calculation, occasionally some more intensive calculation (e.g. model search, smoothing), static and interactive graphics. Statistical systems are seen to be computer workbenches which offer a variety of statistical tools. Quite rightly, much time is spent in statistics courses on understanding when and how to use some of these tools. We train statisticians to be users of the technology.

One might argue, then, that a reasonable objective would be to make our students more savvy consumers. This sometimes leads to survey or comparative courses on statistical packages — information of transient value. The danger of this objective is the notion that we are *consumers*. We need to teach students to be more savvy *producers* of computational resources.

To do this, statistics students must be actively encouraged and rewarded to explore the possibilities of the computer in empirical studies. Statistical contexts should be used to teach basic computational building blocks (e.g. selected algorithms, data structures, graphics, interface, programming principles, . . .). But above all, computational creativity should be actively promoted. In short, training in statistical thinking should be supplemented by training in computational thinking.

A natural place to challenge a student's computational creativity is in the first course on statistical computing. The remainder of the paper discusses one such course which I have given with some success at Waterloo over the past few years.

2 The Plan

As with any educational offering, we need to consider four essential interdependent components of course design: the goals, the content, the audience, and the actual delivery. In this course the goals and delivery transport to different audiences while the details of the content may need adaptation for particular audiences.

The primary goal is for the student to experience computational thinking within the context of statistics. They should come away with a sense of excitement and some confidence about what *they* can achieve with a computer. A secondary goal is the exploration of the interplay between Computer

Science and Statistics — the topics and rigour will depend upon the audience.

The course is delivered in a single 12 week semester,the first 2/3 of which are formal lectures with topics covered fairly quickly. In that time, assignments can be given to encourage students to engage in the course early. The remaining four weeks (or 12 hours of regularly scheduled classes) are given over to the students to work on a large group project. It is important that the project be run entirely by the students; in the final 1/3 of the course the instructor's role becomes that of critic, cheerleader, and information resource.

Like everyone else, students will become actively engaged if given real responsibility and creative opportunity. Responsibility is encouraged in many ways — the use of a *group* project being particularly effective. At the end of term, each student hands in an independent report describing the overall project, the contributions of the individual members, and a more detailed description of their own contribution. Each student is assessed on their written report, their software, and their contribution to the discussion and direction of the project. Giving a clear expectation of activity is also important. I make it clear that I expect students to have worked together or independently between meetings in order to have material for presentation and discussion at the next meeting. I also declare a rarely invoked rule that I leave the meeting if no activity occurs for five consecutive minutes. Creativity can be encouraged by the instructor by recognizing and encouraging students interests. A willingness of the instructor to give impromptu lectures and pointers to resource material (or persons) is particularly helpful in this regard.

Groups of size five to seven seem about right. Our class sizes are small in statistical computing, typically about 7 to 12 students and at most 20. For larger class sizes the number of lectures could be reduced to cover the intersection of topics needed by all groups; other topics would be researched by the students as needed.

As regards our audience, the students are typically senior year undergraduates and have had several terms of work experience. Minimally they will have had 3 courses (a course being 12 weeks of lectures) in calculus, 3 in algebra, 2 in Computer Science, 1 in probability, 1 in statistics generally, and 1 course in applied linear regression. Before graduating they will have had at least 9 further courses in the mathematical and computational sciences. Typically students in this course will have previously taken other statistics courses and/or are taking other statistics courses at the same time. They will have used some statistical package, typically S.

Because it is the only advanced (third year) statistics course guaranteed to have been taken by everyone, the statistical content of the present course builds around applied linear regression. The formal lectures therefore concentrate on computational issues related to regression. These include such traditional statistical computing topics as roundoff error analysis, solving linear systems of equations, and discussion of various matrix decompositions. This discussion allows one to focus on the various calculations needed for least-squares regression and diagnostics (influential and collinearity). The statistical theory is review with the possible exception of the depth of treatment given to diagnostics. For the most part, the computational detail is new to them. The design, use, and critical assessment of statistical graphics is also explored in lectures. The remainder of the lectures are spent on programming principles and practice – data and procedural abstraction, object-oriented programming, as well as details of the programming language being used. Finally, some pointers on software implementation of statistical analysis strategy is given where objects represent steps in an analysis (e.g. Oldford and Peters, 1988). In the most recent offering the impromptu lectures given were on generating random variates, spline-based scatterplot smoothing, and nonparametric regression methods such as additive models and projection pursuit; these topics vary from offering to offering.

The students were assigned the task of organizing and developing an interactive display oriented system for carrying out a linear regression analysis. Throughout they were continually encouraged to think about what *could* be done, to move beyond what they had seen in commercial systems, to imagine what some idealized system should look like. And then to produce one.

2.1 Base software environment

A considerable base of software was available to the students by using the Quail system (Oldford *et al*) which includes:

- the **Common Lisp** programming language including its powerful object system allowing multiple class inheritance and generic functions which can type on any number of arguments.

- a base **programming environment** supplied by the Macintosh Common Lisp vendor (Digitool, 1997) which is quite modern — providing lisp file editors, incremental compilation, program steppers, process backtrace, structure inspectors, and some program analysis tools (who calls, etc.).
  Similar features are available for the PC from Franz Lisp for their Allegro Common Lisp (1997).

- **Quail's multidimensional arrays** including
  — array mapping operators along any dimensions
  — sorting, permuting, ranking along any dimensions
  — usual matrix operators including solution of linear systems
  — matrix decomposition objects (e.g. QR, LU, SVD, . . .)

- **Quail's statistical functionality** including
  Summary statistics — mean, median, percentiles, standard deviation, ...
  Data objects — array objects containing some meta-data information.
  Model objects — Extended Wilkinson-Rogers specification of generalized additive models (e.g. see Chambers and Hastie, 1992). Only the Gaussian linear model class is needed for the course. See Anglin and Oldford (1993) for further detail.
  Fit objects — contains pointers to the model, the data, and results of fitting one to the other.
  Probability objects — classes representing a standard suite of univariate discrete and continuous distributions with each providing a variety of probability calculations and pseudo-random variate generation.

- **Quail's graphic objects.** The model for these was first introduced in Hurley and Oldford (1991) and can be seen in an early demonstration in the video Hurley and Oldford (1988).
  Views and viewed-objects — A graphic in Quail is a data structure called a *view* which can be displayed simultaneously in any number of viewports. The metaphor is that each graphic is a "view" of some other object, its *viewed-object*. Hence every *view* data structure retains a pointer to the viewed object.
  Compound views — views which contain subviews. Compound views position their subviews in a display. The compound view and every subview may have its own viewed-object; subviews can themselves be compound views.
  Stock statistical graphics — dotplots, boxplots, histograms, stem and leaf, 2 & 3D scatterplots, 2 & 3D function plots, scatterplot matrices, brushing, linking, ...
  Controls — needle-sliders, bar-sliders, push-buttons, editable text-input, dialogs, pop-up menus. These could operate on anything.
  View layouts — compound views which layout subviews in row, column, or grid fashion, or more generally at arbitrary positions specified by the user.
  Interactive display — every view responds to three mouse buttons (left, middle, right) alone or in combination with two modifier keys (shift and ctrl). Unmodified mouse buttons typically produce menus which refer to the physical display of the selected view; ctrl-mouse buttons refer to the *viewed-object* of the selected view.

- Two **strategic functions** having methods for any object
  (*Signposts* object ...) — returns a list of "signposts" particular to the given object; each signpost is a kind of control button view which if displayed and mouse selected would lead to some other relevant display peculiar to that signpost from that object.
  (*Display* object ...) — returns a view, which if drawn would produce a reasonable display of the given object. Display always accepts a boolean argument :signposts? which if true will return a view augmented by signposts.

  Ctrl-middle-mouse on any view pops a menu offering the user the opportunity to call display on the viewed-object, with or without signposts. This means from a display, any viewed-object could be interacted with directly.

Other object oriented systems allowing user defined classes and generic functions could also be used. The critical base language features are the ability for the user to define classes having inheritance, generic functions, and to specialize system defined methods. Programmatic construction, display, and manipulation of statistical graphics is also essential.

3 Project development

The formal lecture part of the course included laboratory sessions where the students explored the programming environment first hand. Examples of code fragments were provided to illustrate some points in the lectures. Nevertheless, virtually no progress was made on the project until the formal lectures ended. Once I sat and listened, saying little more than "So, what are you going to do?" the project began in earnest.

3.1 Group dynamics

The students knew each other only casually (in spite of the small class size) and so were initially reluctant to toss ideas out in front of the group. Moreover, there was a natural tendency for each person to address their remarks to the instructor. Rather than answer, I would turn the question over to someone else for comment. As the students became more responsible for the material, they became more self-reliant. At the beginning of each class I would ask "What have you done since I last saw you?" thus encouraging the students to meet regularly outside the class. I saw my role as that of a constructive and critical resource for their self-directed research. By the end, the group was cohesive, self-reliant, and without exception willing to pursue self-learning aggressively.

3.2 Workload pattern

Initially there was a great deal of joint work as the project began to be fleshed out. The enormity of the project meant that it would need to be broken into pieces digestible by individuals or pairs of students. This also affected the project design — it had to be modular. Each piece required independent study. During this phase students would report what they had learned in their separate readings. Sometimes detail would be discussed only with me, or with me and those one or two others whose own piece of the project would be affected. Some things, like program design, could interest or affect everyone and so saw broader discussion. Putting it all together required further communication between students whose pieces interfaced one another's. Project write-up was each individual's responsibility.

3.3 Content development

At the start, students seemed to view regression analysis as a rather flat structure having little texture. Essentially, their recollection seemed to consist of an unordered collection of possible methods — model search, residual plots, anova, . . . — each student giving prominence to one or two of these and largely ignoring the rest. It was as if one or two pieces of their previous course had stuck with them regardless of its importance or merit (e.g. one student would describe how we needed forward selection, and then a little later "stepwise regression"). No strategic sense was present at all.

At the end of the first session, I suggested they come to the next session prepared to identify a sequence of steps which might be taken in a regression analysis. The result was a flow chart describing a process for regression analysis. It was relatively linear and prescriptive.

"What if" questions soon showed the linear structure to be too constraining. The overly prescriptive nature became apparent when the designers imagined themselves as the users — the putative user for whom the design had been created turned out to be non-existent. The structure was continually repaired after each assault and quickly became quite complicated containing, for example, feedback loops and many steps which could jump to a great many others. Abstractions fell away, crowded out by detail. Any sense of strategy seemed lost.

Having explored both ends of the spectrum, it seemed wise to set aside some general task areas, determine when they might be undertaken in an analysis, and ask what information should be available at that step in order to carry out the tasks. Isolating task areas and following the data led to the final strategic representation of a regression analysis shown as the directed graph of Figure 1. Each node is a step, and each arc indicates *potential* movement from one step to another. (The arc from the "model search" node carries a question mark and leads nowhere because limited progress was achieved in this area.) The strategy is loose in the sense that there can be a number of activities undertaken at each node that do not involve movement to new nodes and that it is perfectly possible, given the appropriate data structure, to enter the strategy beginning at *any* node.

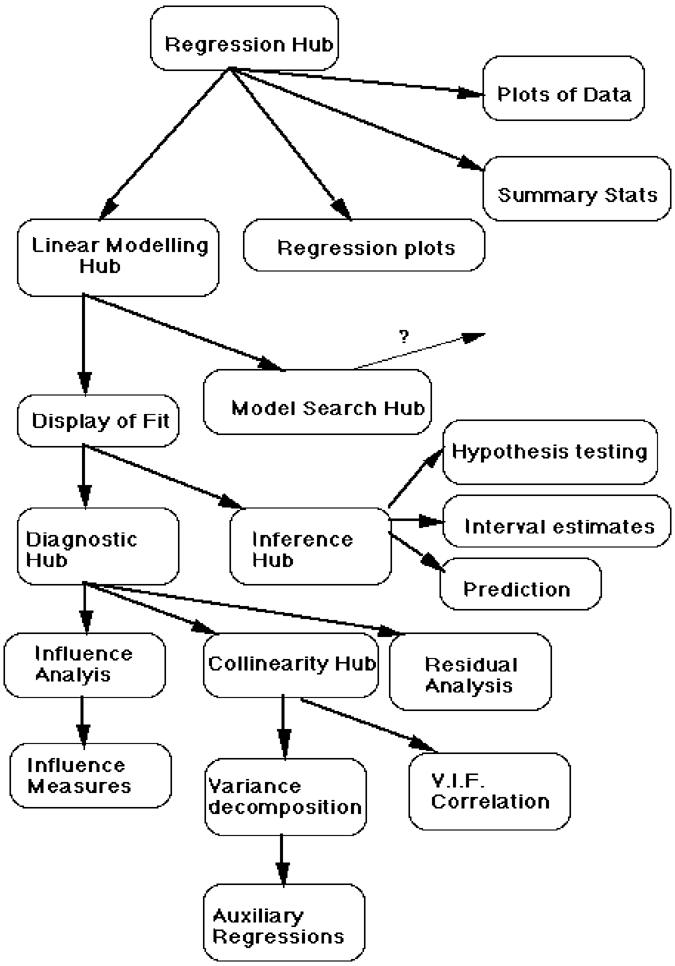

Figure 1: Students' strategy for regression analysis.

3.4 Implementing the strategy

The nominal entry point for regression analysis is the topmost node, the *Regression Hub*. Entry means that a dataset has been identified and that the intention to pursue a regression analysis on it declared. This produces the interactive window seen in Figure 2. It displays some information on the data (variate names, case names, dataset name) and some buttons at the bottom. Each of the first three buttons lead to one of the next nodes in the strategy — here to the *Linear Modelling Hub*, to *Summary Stats*, or to *Regression plots* respectively. The plotting buttons prompt the user to choose among a wide variety of plots available for any dataset. Each column refers to one- two- and multiple-variate plots; the second row (which would on a screen read "Plot by") produces a table of plot whose cells are arranged according to the values of one or two other variates.

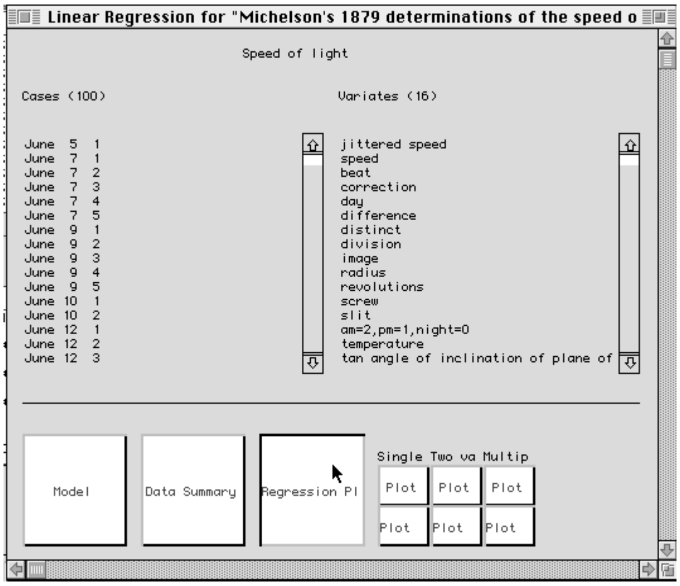

Figure 2: Regression Hub

The construction of the *regression-hub* illustrates the modularity of the code. Two major pieces of the display are available directly from the underlying dataset. Calling the generic function *display* on the dataset without :signposts? returns a *compound-view* displaying the dataset information shown in Figure 2. A separate call to the generic function *signposts* on the dataset produces the collection of plot buttons — this is why no separate display is produced corresponding to the stratefic node *Plots of Data* in Figure 1. With the data display and the data signposts in hand, the designer of the *regression-hub* need only construct the three buttons leading to new nodes and lay out all of these in an appropriate display as in Figure 2. The *viewed-object* of the resulting layout is a *regression-hub* object and conversely the display of Figure 2 is the result of calling *display* on a *regression-hub* object — the collection of buttons are its signposts.

Upon entry, each node in the strategy network produces a new window containing an interactive *compound view* that displays relevant information. For most nodes, the *compound view* offers the user the possibility of changing the display to show other relevant features of the underlying information. Figure 3 shows the display associated with the *Regression plots* node. This would appear if, as shown in Figure 2, the "Regression Plot" button was selected from a *regression-hub* display. Various polynomial fits can be added to the display (through the buttons) and the variates displayed can be changed (through pop-up menus available by direct mouse interaction with the plot itself).

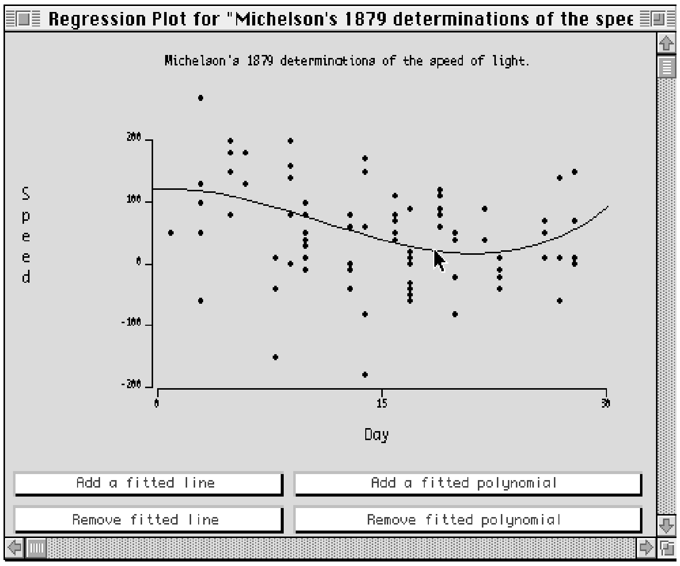

Figure 3: A "regression plot" showing a cubic added to the plot. Selecting the curve (with "CTRL" pressed as well) pops up a menu that allows the user to display the underlying fit-object with signposts.

Note again the loose and coarse nature of the strategy. The node that displays a fit and which leads to diagnostics and inference can be reached from a signost in the *Linear Modelling Hub*. But it can also be reached through any *view* that has a *fit-object* as its *viewed-object*. For example selecting the curve in Figure **??**, the user can choose to have the underlying fitted cubic displayed as if it had been reached through a *Linear Modelling Hub*. Viewed objects, the strategic functions *display* and *signposts* are critically important to designiong and creating a flexible strategy.

Some node displays, like the *Regression Hub*, contain buttons which lead directly to the next nodes in the strategy — hence the names *hub* for the display, and *signposts* for buttons of this type. Nodes which have no signposts are said to be *terminal* in the strategy. Terminal nodes may be extremely interactive as, for example, in the case of the *Regression plots* node; they just do not provide signposts that lead to new nodes. Any number of nodes may be displayed in separate windows at the same time.

The structure of each node breaks into three distinct descriptive pieces: initial information displayed, activities available, and signposts. Aside from the *Regression Hub* and the *Regression Plots* node already discussed, the characteristics of the remaining nodes from Figure 1, as implemented by the students, are described in the appendix.

4 Summary of experience

The experience, though worrying at times, has been overwhelmingly positive. Through their course evaluations and conversations afterwards, the students tell me that they appreciated working on a complex interdependent project. The team and active learning approach was universally praised. They were delighted to have an instructor cover topics as they chose. Each learned much from their independent studies. The principal negative comment was that they had not started the project soon enough.

Most interesting was the common remark that they came to better understand statistical topics that they covered in other courses. This latter is due, I believe, to the fact that they were responsible for developing a statistical strategy. It forced them to review and synthesize previous material and to explore the boundaries of the strategy. This naturally led to wide ranging statistical discussion. Moreover, there was a dawning recognition that although details differ from area to area that perhaps the following structure was common to many areas of statistical analysis:

- Analysis session
- Graphical and numerical summaries
- Model specification and editing
- Model Search
- Diagnostics
  - Influence
  - Identifiability
  - Model checking
- Inference

Covering two or more areas could lead to interesting discussion as to how one might capitalize on this (or some other) common structure.

The group project was essential in having the students experience computational thinking. They designed, developed, and debugged software structures that dealt with routine statistical calculations (e.g. diagnostics, hypothesis tests), relatively complex statistical algorithms (e.g. leaps and bounds model search, lowess smoothing), new interactive statistical graphics (e.g. regression plots, plots of $h_{ii}/(1-h_{ii})$ versus t_i), interface design (e.g. every node in the strategy), object-oriented design of the classes and functions associated with the various analysis hubs, and the overall strategy of carrying out a linear regression analysis. While each student focussed on an area in the strategy, the breadth of the area and the detail required in implementation ensured that each student worked on statistical and computational issues at a variety of abstraction levels. The creativity of the students is easily seen in the results.

Appendix

Here we describe the student designed structure implemented for each node shown in Figure **??**. Excluded are those nodes uncompleted and those nodes described in the text above. The structure of each node is separated into three distinct descriptive pieces: initial information displayed, viewed-object, activities available, and signposts.

Summary Stats

Initial display: Scrollable table of summary statistics for each variate in the dataset.
Viewed-object: The dataset.
Activity: None.
Signposts: None. Terminal node.

Linear Modelling Hub

Initial display: Dataset summary as in the regression hub and two editable text fields – one for entering the name of the response variate, the other for entering the linear predictor in extended Wilkinson-Rogers notation.
Viewed-object: Linear-modelling-hub
Activity: Editing the model formula to be used by the fit button.
Signposts: 1. "Model Search" leading to a corresponding hub (unimplemented) and 2. "Fit" which fits the specified model via least-squares and calls display with signposts on the resulting fit-object.

Display of Fit

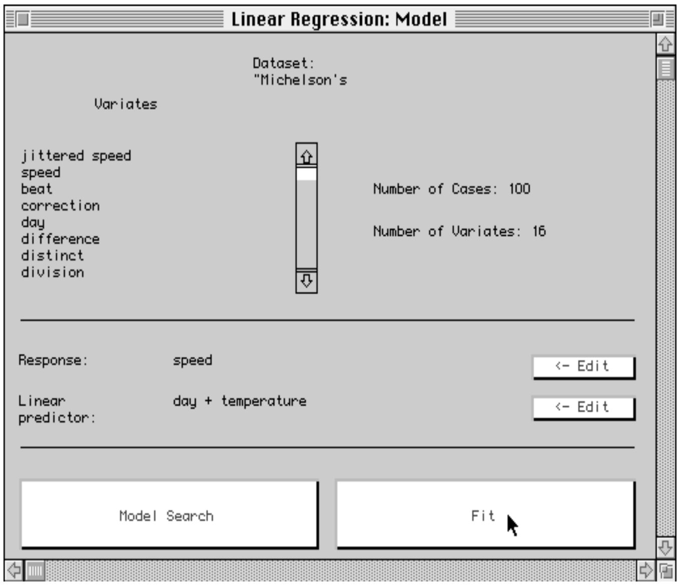

Figure 4: Modelling hub

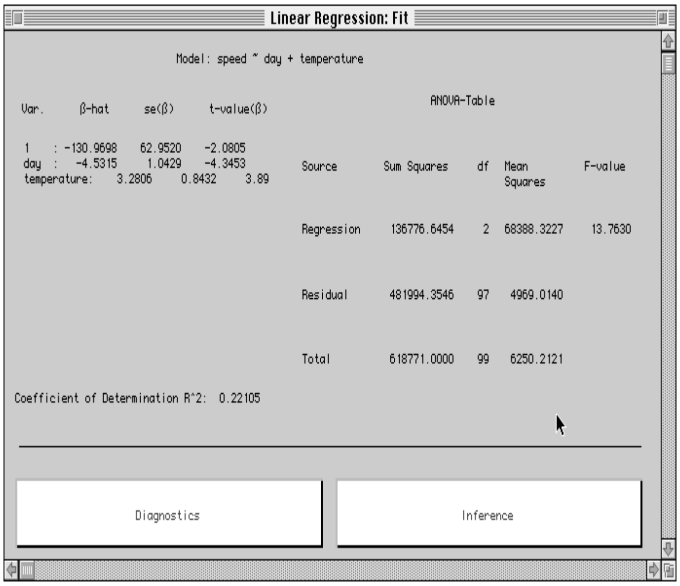

Figure 5: Display of fit

Initial display: Two tables 1. the estimates, standard deviations, t-statistic for each term in the model and 2. the Anova table including F-statistics and the corrected R^2 statistic.
Viewed-object: The least-squares fit-object which itself contains pointers to a model-object, the dataset, and calculational details like a QR-decomposition object. The whole hub is simply the result of calling display with signposts on the fit-object which is of interest because fit-objects can appear as viewed-objects of many other views (e.g. the curve displayed in a regression plot).
Activity: None.
Signposts: 1. "Diagnostics" and 2. "Inference"

Diagnostic Hub

Initial display: Model formula. Boxplots of each variate in the dataset.
Viewed-object: Linear-regression-assessment object which itself contains pointers to the fit-object.
Activity: None.
Signposts: 1. "Influential Analysis", 2. "Collinearity Analysis", and 3. "Residual Analysis"

Influential Analysis

Initial display: Table containing case name, h_{ii}, least-squares residual, studentized residual, and externally studentized residual. The table is scrollable over the cases.
Viewed-object: The fit-object.
Activity: Buttons to replace table display by 1. h_{ii} vs. i and $\frac{h_{ii}}{1-h_{ii}}$ vs. t_i (the externally studentized residual), 2. t_i vs. i and $\frac{h_{ii}}{1-h_{ii}}$ vs. t_i, and 3. Back to the table display.
Signposts: 1. "Normed Influence Measures"

Normed Influence Measures

Initial display: Table showing the mathematical definition of four normed influence measures as described for example in Cook and Weisberg (1983) including "Cook's distance" and "DFFITS" of Belsley et al (1980).
Viewed-object: The fit-object.
Activity: Four buttons, one for each normed measure in the table. Pressing a button produces an index plot of that measure to the right of the table.
Signposts: None.

Collinearity Diagnostics

Initial display: None
Viewed-object: collinearity-hub.
Activity: None.
Signposts: 1. Variance decomposition proportions, 2. Variance inflation factors, and estimators' correlation matrix, 3. Signal to noise testing.

Variance decomposition

Initial display: 1. Table showing, the log_{10} condition indices and a variance decomposition proportions table (see Belsley et al 1980, Belsley 1991).
2. A scrollable list of the terms in the model.

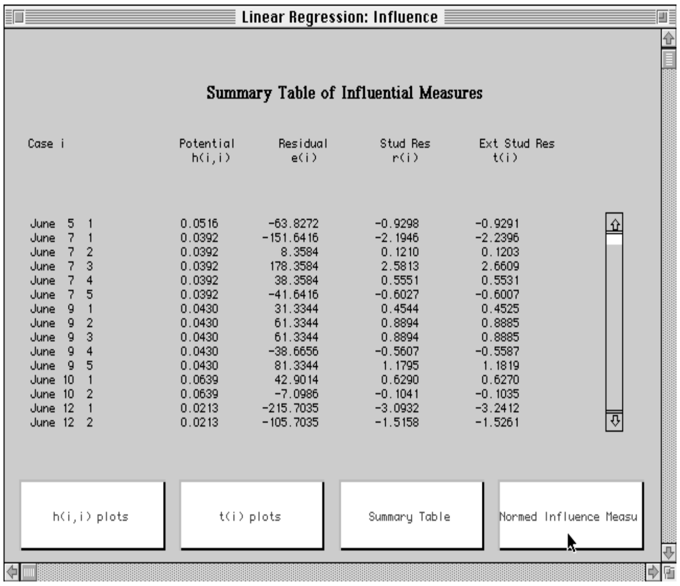

Figure 6: Influence analysis

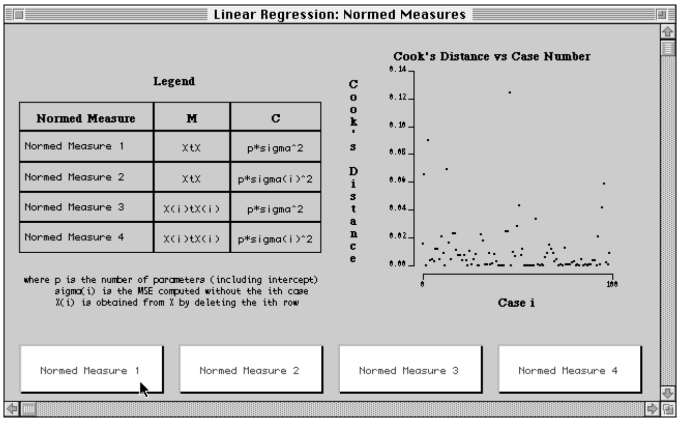

Figure 7: Normed Influence measures

3. An editable list identifying the terms to be considered as the dependent variate in an examination of auxiliary regressions (see Belsley 1991 for strategy).
4. A list of the auxiliary regression models already determined.
Viewed-object: variance-decomposition object.
Activity: Editing of the terms to be considered dependent in an auxiliary regression analysis.
Signposts: 1. Fit the identified auxiliary regressions.

Auxiliary regression analysis

Initial display: 1. A list of all fitted auxiliary regression models having the each specified term as the dependent variate.
Viewed-object: auxiliary-regression-analysis.
Activity: Fitted models may be selected.

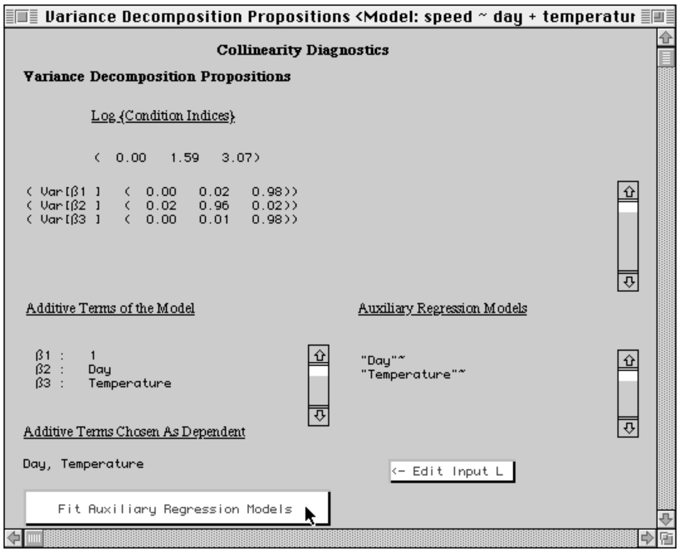

Figure 8: Variance decomposition proportions

Signposts: 1. Fit from the currently selected model can be displayed with signposts

V.I.F. and correlation

Initial display: 1. The correlation matrix for the coefficient estimators.
2. The variance inflation factors associated with each coefficient.
Viewed-object: vif-correlation-analysis.
Activity: None.
Signposts: None.

Residual Analysis

Initial display: None.
Viewed-object: The fit-object.
Activity: Offers three different types of residual examination. 1. Gaussian qqplot, 2. Residuals vs. fitted values, and 3.Side by side of (a) residuals versus an explanatory variate in the model and (b) Added variable plot for the effect of the same explanatory variate. Separate buttons produce each of 1,2, or 3 in the display area. A fourth button “next” cycles the side-by-side plots of 3 through the explanatory variates in the model one at a time. Plot 3(b) has a “lowess” smooth superimposed.
Signposts: None.

Inference Hub

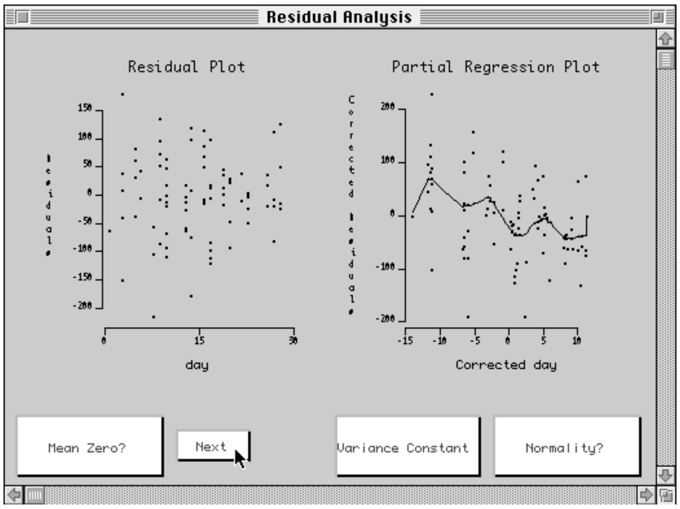

Figure 9: Residual analysis

Initial display: None.
Viewed-object: inference-hub.
Activity: None.
Signposts: 1. Hypothesis tests, 2. Confidence Intervals, 3. Prediction Intervals.

Hypothesis tests

Initial display: 1. Table of estimates, standard errors, t-statistics, and significance levels for each coefficient in the regression.
2. Table of results for testing a general linear hypothesis.
Viewed-object: The fit-object.
Activity: 1. General linear hypothesis of the form $A\ \beta = c$ can be tested. User is prompted for the matrix A and the corresponding vector c. Resulting F-statistic and significance level are then displayed.
2. Selecting terms in the teble of estimates can be tested for simultaneously being zero. Results are displayed in terms of the corresponding general linear hypothesis.
Signposts: None.

Confidence Intervals

Initial display: Table of estimates, standard errors, and 95% confidence intervals for each coefficient in the regression.
Viewed-object: The fit-object.
Activity: 1. Change the confidence level.
2. Produce a confidence interval for an arbitrary linear combination of coefficients.
Signposts: None.

Prediction Intervals

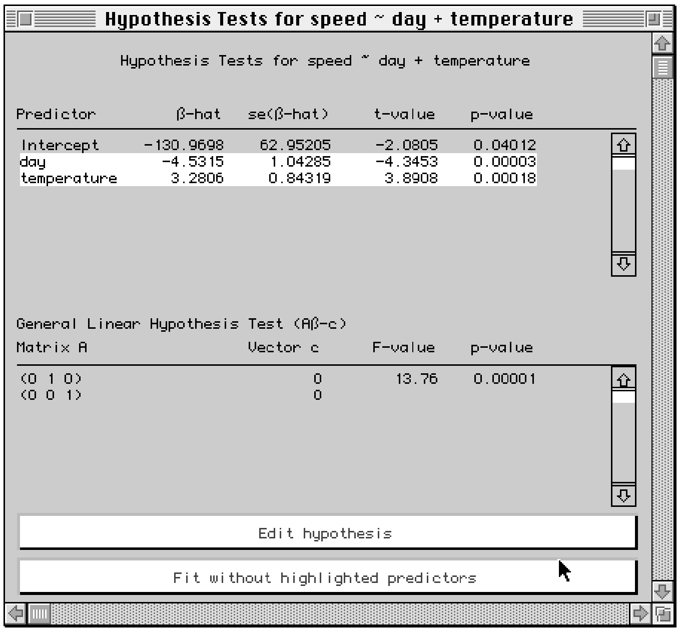

Figure 10: Hypothesis tests

Initial display: Table of estimates, standard errors for each coefficient in the regression. A 95% prediction interval and a point prediction for the response at the pre-specified values of the explanatory variate (prompted for at creation).
Viewed-object: The fit-object.
Activity: Change the prediction level.
Signposts: None.

References

Allegro Common Lisp (1997), PC and Unix based Common Lisp from Franz Lisp Inc, Berkeley California.

Anglin, D.G. and R.W. Oldford (1994), "Modelling Response Models in Software" pp. 413-424 in *Selecting Models from Data: A.I. and Statistics IV* edited by P. Cheeseman and R.W. Oldford, Springer-Verlag Lecture Notes in Statistics, 89.

Belsley, D.A. (1991). *Conditioning Diagnostics: Collinearity and Weak Data in Regression* Wiley & Sons.

Belsley, D.A., E. Kuh, and R.E. Welsch (1980) *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity* Wiley & Sons

Cook, R.D. and S. Weisberg (1983) *Residuals and Influence in Regression*

Hastie, T. and J.M. Chambers (1992), (editors) *Statistical Models in S* Wadsworth Publishing.

Hurley, C.B. and R.W. Oldford (1988). "Hierarchical views of statistical objects" *STAT 88-19* video Statistics technical report from the University of Waterloo. Also available from the video library of the *ASA* sections on *Statistical Computing* and *Statistical Graphics*. 19 minutes VHS (NTSC & PAL).

Hurley, C.B. and R.W. Oldford (1991). "A software model for statistical graphics" pp 77-94 of *Computing and Graphics in Statistics* edited by Andreas Buja and Paul A. Tukey IMA Series on Mathematics and its Applications, Volume 36.

Macintosh Common Lisp (1997), Digitool Inc., Cambridge Massachusetts.

Oldford, R.W., C.B. Hurley, D.G. Anglin, M.E. Lewis, and G.W. Bennett (1988-1997) *Quail: Quantitative Analysis in Lisp*. A statistical programming environment available free of charge from R.W. Oldford at the University of Waterloo.

Oldford, R.W. and S.C. Peters (1988). "Towards more statistically sophisticated software." *SIAM Journal for Statistical and Scientific Computation*

Steele, G. (1991), *Common Lisp: The Language, 2nd Edition* Digital Press.