Dynamic Graphics and Regression Diagnostics using XLISP-STAT

R. D. Cook and S. Weisberg
Department of Applied Statistics
University of Minnesota
St. Paul, Minnesota 55108*

Technical Report No. 565
October 7, 1991

Abstract

A set of additions to the standard `regression-model-prototype` in LISP-STAT (Tierney, 1990) are described. These additions do all of the graphics described in Cook and Weisberg (1989a), as well as a number of other graphics (e.g., Cook and Weisberg, 1983, 1989b, 1991). A set of other useful functions is also included. The code for these additions is available in the XLISP-STAT library in `statlib`.

1 Introduction

The LISP-STAT environment designed by Tierney (1990) provides an excellent platform for dynamic graphics in statistics. We have used this program as a basis to implement the ideas for regression diagnostics that appear in Cook and Weisberg (1989a), hereafter called "CW89". This document describes our code, gives an indication of how the code works, summarizes what the code does, and provides help on getting started. To use the code, you will need to have a copy of XLISP-STAT. In this document, we assume that this program is available to you. We have tested the code on Unix workstations and on Macintosh II computers; we have no reason to believe that it will not work with the Windows version of XLISP-STAT.

We are distributing this code for several reasons. First, we believe that the methods provided here can be very helpful in regression modeling. The methods are difficult or impossible using standard programs. Second, we show how XLISP-STAT can be used to implement new statistical ideas. Third, we provide code that may be helpful to others who are interested in developing similar methodology.

These enhancements to the `regression-model-proto` are available without charge, and may be freely distributed or modified. However, they are not to be sold for a profit, or offered as an inducement to purchase a commercial product. In general, this code may be distributed using the same rules for distributing XLISP-STAT. Although we have attempted to produce bug-free code, errors surely remain. We would be glad to hear about bugs: send e-mail to *sandy@umnstat.stat.umn.edu*.

1.1 Using statlib To Get the Code

The code is available in the XLISP-STAT library on `statlib`. The get the code over the internet, send the message:

*This work was supported by National Science Foundation Grant number 9001298. We would like to thank Luke Tierney, Jim Harner and Nate Wetzel for their help with this project, and Julian Faraway for useful comments.

```
mail statlib@lib.stat.cmu.edu
Subject:
send dyndiag from xlispstat
```

Save the information that is sent back to you in a file, and follow the instructions included for decoding. When unpacked, you will have a LaTeX copy of this document called `doc.tex`, and several files whose names all end in `.lsp`. Put all of the .lsp files into the directory (or, on the Macintosh, folder) from which you launch XLISP-STAT.

1.2 Distribution by *ftp*

The code may be obtained via anonymous *ftp* to the machine *umnstat.stat.umn.edu*. Here is an abridged transcript of a session that will get the code:

```
your-machine% ftp umnstat.stat.umn.edu
Connected to umnstat.stat.umn.edu.
Name (umnstat.stat.umn.edu:): anonymous
Password (umnstat.stat.umn.edu:anonymous): (use your e-mail address)
ftp> cd pub
250 CWD command successful.
ftp> get dyndiag
200 PORT command successful.
150 Opening data connection for dyndiag (ascii mode) (201290 bytes).
226 Transfer complete.
local: dyndiag remote: dyndiag
205792 bytes received in 11 seconds (18 Kbytes/s)
ftp> quit
```

The file `dyndiag` is a Unix shar file, identical to the one that could be obtained via *statlib*.

1.3 Distribution by Disk

If you have the code on a floppy disk, copy every file on the disk that ends with `.lsp` into the directory (folder on the Macintosh) from which you launch XLISP-STAT. The remaining file, `doc.tex` is a LaTeX copy of this document.

1.4 Getting Started

When XLISP-STAT starts, it looks for a file called `statinit.lsp` in the current directory or folder. If such a file exists, it is loaded by the system. To load the dynamic diagnostics code automatically, modify your `statinit.lsp` file, or create a new one, by adding the following line:

```
(load "cwinit")
```

To load the code manually, enter the above command in the listener window (which is the name of the window for typing commands or getting output), or, on the Macintosh, select "Load" from the "File" menu, and, in the dialog box that will appear, select "cwinit.lsp." Included are most of the data files used in CW89 (cars.lsp, cloud.lsp, logistic.lsp, rat.lsp, lathe.lsp and trees.lsp), and these can be used to reproduce the graphs discussed in the paper.

To get a quick overview of the enhancements, enter the following command:

```
(load "trees")
```

This will create a regression object called `tree-reg`, and this object will be sent the `:graphics-menu "Tree-data"` message. A menu called `Tree-data` will appear in your menu bar. You can try out the items in this menu as a test of the features available.

2 Regression Methods

`Regression-model-proto` is the base prototype for fitting regression models using LISP-STAT. Included here are many additions to the original version written by Luke Tierney, including most of the dynamic diagnostics proposed by CW89 as well as a few other graphs and some new "standard" features such as easy case deletion. The methods described below are not in the standard distribution of regression-model-proto. For a description of the standard methods, see Tierney (1990. Sec. 6.5.2).

2.1 Calling sequence

The model is initialized with a function call as described in Tierney (1990, p. 201). Here is an example of the function call, assuming the data is in a plain text file called "tree.dat" with 3 columns in the current directory[1]:

```
(def tree (read-data-columns "tree.dat" 3))
(def tree-reg (regression-model (select tree '(1 2 ))
                                (select tree 0)
                                :predictor-names '("Diameter" "Height")
                                :response-name "Volume"))
```

All of the methods described here are then accessed by sending messages to the object `tree-reg`. For example, to get the graphics menu to appear, send the message:

```
(send tree-reg :graphics-menu "Tree-data")
```

To get an ARES plot, enter:

```
(send reg :ares)
```

or select the "ARES plot" item from the `Tree-data` menu.

2.2 Routines used

This code uses standard XLISP-STAT routines, plus bits and pieces of ALL the code provided. Thus, you must load EVERYTHING for the code to work properly. Besides additions to regression-model-proto, there are additions to graph-proto and its descendants, a new prototype called `project-proto` that computes a QR factorization, and allows accessing both the factorization and various related quantities, a large number of plot overlays (see Tierney, 1990, p. 285), and several functions collected in `fun1.lsp`. Some of these functions may be useful in other circumstances, and they may be freely used elsewhere.

2.3 Computational comments

Regression-model-proto uses the sweep algorithm is its basic computational tool. For some methods, such as ARES plots, sweep is not a convenient paradigm for computations, and the QR factorization provides a friendlier environment. As a result, ARES and a few other methods use the QR factorization via the project prototype. Rewriting all of regression-model-proto to be QR based would not be very hard.

[1] On Unix systems, the current directory is the same as the directory from which XLISP-STAT is started. On the Macintosh, the current directory or folder is the last folder accessed. For example, if a file in the folder "Datasets" is opened for editing using the built-in editor, then the current folder is "Datasets."

2.4 Plotting methods

Most of the plots created by sending a message to a regression model proto can later be accessed and will be updated whenever the regression model is updated. In particular, when a case is deleted or restored, the regression model is recomputed, and most plots are redrawn. Most plots have two additional menu items added to their standard menu. These menu items, which appear at the end of the menu, can be used to delete or restore cases to a model from the plot. Points not used in computations will generally appear as selected points on a plot, but of course they can be marked as desired in the usual ways. Where there are points not used in calculations, one of the additional menu items is "Select Deletions", so these points can always be identified.

In addition, all plots include a number of plot controls, buttons and sliders that can do useful things to the plot. Almost all of these are self explanatory, but see also the section on Plot Controls.

- `:plot (&optional x-axis y-axis z-axis &key (plot-controls t))` makes plots. If no specification is given, the default is a scatterplot of 'fit-values *versus* 'studentized-residuals. If only x-axis is specified, a histogram is drawn. If two axes are specified, a scatterplot is drawn. If three axes are specified, a spin-plot is drawn. If :plot-controls is t, the default, then many appropriate control buttons, sliders, and so on are added to the plot. The quantities to be plotted can be either a list or vector of numbers, or else they can be a function that when funcalled will produce a list of numbers to be plotted. If the latter procedure is used, then the plot will be automatically updated whenever the underlying regression model is changed, for example by changing the 'included slot, thereby deleting or restoring cases, or by transforming a predictor or the response. If one of the names supplied below (they all begin with a single quote `'`) is used for an axis specification, then a function is created that when funcalled will compute the values specified.
  - 'response (response variable)
  - 'yvar (response for glims or regression models)
  - 'y (response for regression models; working variate for glims)
  - 'raw-residuals (e_i)
  - 'residuals ($\sqrt{w}e_i$)
  - 'studentized-residuals ($e_i/\hat{\sigma}(1-h_{ii})$)
  - 'externally-studentized-residuals ($e_i/\hat{\sigma}_{(i)}(1-h_{ii})$)
  - 'jackknife-residuals (same as externally studentized residuals)
  - 'residuals-squared (see discussion under Non-graphical methods below)
  - 'leverages (h_{ii})
  - 'cooks-distances
  - 'distances (same as Cook's distance)
  - 'sqrt-cooks-distances
  - 'fit-values
  - 'case-numbers
  - 'signed-local-inf (direction of maximum curvature; see discussion under Non-graphical methods below)
  - 'local-inf (absolute values of direction cosines)
  - 'normal-scores (calls the quantile method above with no arguments)

  If you want to plot against a column of X, you may specify a column number, or its quoted predictor-name (remember that Lisp starts counting lists with element zero, not element one).

To plot some other list of data, you can just give the data or its name. This method automatically determines axis-labels and a plot title, but these can be modified in the usual way by sending the graph the appropriate messages. The apostrophe before the names shown above is required. The method `:return-function` takes the keywords given above and turns them into function. For example, if you give the specification `'residuals`, then the following function is created: `#'(lambda () (send self :residuals))`. Through this device, if the specification of a model is changed (for example, if a point is deleted), then a plot can be redrawn by executing the function that specifies the data. You can supply any other function.

Typical calls to plot might be:

```
(send reg :plot 'case-numbers 'local-influence)
(send reg :plot 0 'y 1)
(send reg :plot #'(lambda () (* (1- (send reg :leverages))
                                (send reg :fit-values)))
                #'(lambda () (^ (send reg :raw-residuals) 2)))
```

The first plot is of {case numbers, direction of maximum local curvature for case weight perturbations}. The second plot is a 3-D plot of {first predictor, response, second predictor}. The third plot is of $\{(1 - h_{ii})\hat{y}_i,\ e_i^2\}$, a plot for heteroscedasticity. All three plots will be correctly updated if cases are deleted because the axes are specified by functions that create the data, not the data themselves.

- `:scatterplot-matrix (arguments)` This message draws a scatterplot-matrix, using the same arguments as the `:plot` command discussed above. The default plot is a scatterplot matrix for all the predictors and the response.

- `:qq-plot (&keywords)` produces a quantile-quantile plot of studentized residuals. To change the quantity plotted, set the keyword `:x` by specifying a function or one of the keywords listed under the :plot method above. For example `:x 'y` will give a normal probability plot of the response. To change to a different distribution than the normal, set the keyword `:quantile-function` to the name of the function that will produce the desired quantiles. The function passed to qq-plot must be of a single argument, a number between 0 and 1, as is the case for the normal-quant function. LISP-STAT has a number of other built-in quantile functions, including cauchy-quant, f-quant, poisson-quant, chisq-quant, beta-quant, and t-quant, but most of these require two or more arguments. It is therefore necessary to define your own quantile function that fixes the values of the other arguments. For example, to get a χ^2_5 probability plot of the response in a regression object reg, you can type the following expression:

```
(send reg :qq-plot :x 'y
            :quantile-function #'(lambda (p) (chisq-quant p 5)))
```

An additional function called hnorm-quant in fun1.lsp computes quantiles of the standard half normal distribution. Thus, to get a half normal probability plot of the square root of Cook's distances, as suggested by Atkinson (1985), type

```
(send reg :qq-plot :x 'sqrt-cooks-distances
            :quantile-function #'hnorm-quant)
```

The final keyword is `:plot-controls` which, if set to its default value of t, adds control buttons to the plot. Setting `:plot-controls nil` leaves the control buttons off (this would usually only be desirable if the plot is to be saved for later printing and publication). If any cases in the data are not used in the calculations (that is, the corresponding element of `(slot-value`

`'included)` is nil), then they are not used to compute quantiles. These deleted points will be plotted just to the left of the included data. This is non-standard, and some users may wish to change this to something else.

- `:ares ()` produces an *A*dding *RE*gressors *S*moothly or *ARES* plot. The user will be given dialog boxes to choose the base model and the model to be added, and the method for adding (sequential or as a group). See CW89 for details of what this plot does, and how the calculations are done. The dialog boxes use the `select-list-dialog` function included in fun1.lsp.

  For example, to reproduce Figures 4 to 7 of CW89, p. 281, one first needs to load the lathe data:

  ```
  (load "lathe")
  ```

  This will create a regression object `lathe-reg`, and send it the :graphics-menu message. One can then either type the command

  ```
  (send lathe-reg :ares)
  ```

  or select ARES from the menu. In either case a dialog box appears. In the left scrolling window will appear the names of all the predictors in the model. Double clicking on a name moves it to the right list. To reproduce Figure 4, double click on "Speed", and then click on "OK". A second dialog box appears. Double click on "Feed", and then click on "OK". The plot will now appear, and the animation is carried out be clicking on the slider on the plot.

  To get Figures 5 to 7, start a new ARES plot by typing its message or selecting it from the "Lathe-data" menu. In the first dialog box, select both "Speed" and "Feed", and then "OK". In the second dialog box, select "Speed^2", "Feed^2", and then "Speed*Feed", in this order, and then "OK". A third dialog box appears, giving a choice between adding the variables as a group (all at once), or sequentially; select sequentially, and then "OK". The plot will appear with a slide bar. The index on the slide bar tells you which predictor is currently being added. For example, if the value on the slide bar is 1.5, then predictor number one in the list "Speed^2", "Feed^2", "Speed*Feed", or "Feed^2" since *lisp* starts counting at zero, is being added, and the current value of λ is .5. Figure 5 corresponds to $0 \leq \lambda \leq 1$; Figure 6 to $1 \leq \lambda \leq 2$, and Figure 7 to $\lambda \geq 2$.

- `:avp ()` produces an added variable plot. Variable selection is done via dialog boxes. Buttons appear on the plot to show a contour of constant local curvature using two perturbation schemes, case weight perturbation and predictor perturbation. The details are given by Cook (1986) and Cook and Weisberg (1991). For both perturbation schemes, the contour is drawn at a value of $\sqrt{.33} = .58$ (the total length of the perturbation direction vector is 1, so a point that lies on or beyond the displayed contour accounts for at least 33% of the total length). Detrended avps, obtained via a button on the plot, are discussed in CW89 (Sec. 4.3).

- `:spin-residuals ()` produces a 3-D residual plot $\{P_1Y, (I-P)Y, P_{2.1}Y\}$ where P is the projection on the full model, P_1 is the projection on the base model, and $P_{2.1}$ is the projection on the added variables after adjusting for the base model. Dialog boxes as in :ares are used to get the specification. Details of the plot are given in CW89 (Sec. 5).

  To reproduce Figure 14 of CW89 enter the command:

  ```
  (load "cloud")
  ```

  and then select the "Spin Residuals, 3-D" item from the "Cloud-data" menu (or enter the command from the keyboard). The first dialog box asks for the base model, which consists of all the predictors except "Action". A second dialog box asks for the added predictors, consisting here of just "Action". This will produce the figure in the paper.

- `:avsp` Produces an added variables plot (CW89, Sec. 4.4). The model is specified via dialog boxes. For example, Figure 12 in CW89 can be reproduced by selecting the "AVsP – 3-D" item from the "Cars-data" menu after typing

  ```
  (load "cars")
  ```

  The first dialog box allows choice of a "Base model", corresponding to X_1 in the paper. Select "Weight" and then "OK". A second dialog box appears to choose X_2. Select "Horsepower", and then "OK". A final dialog box is for X_3. Select the remaining predictor "Displacement", and then "OK". Figure 12 is three static views of this resulting plot.

  Work reported in Cook and Weisberg (1989b) suggests that in this, and in all higher dimensional residual plots, the quantities plotted on the axes should be orthogonal to maximize resolution in the plot. An efficient way to do this is the change the quantity on the out-of-page axis to be the part of the second added variable that is orthogonal to the base model and the first added variable. This can be done in this plot by selecting the "Orthogonalize Axes" button on the plot.

- `:dynamic-rankits (arguments)` This does several regressions of $Y^{(\lambda)}$ on the predictors, for λ in the range $(0, 1)$ for the Box-Cox modified power family of transformations. A display is then given of a confidence curve (Cook and Weisberg, 1990) for λ based on the profile log-likelihood. In addition, a window is opened with a rankit (normal probability) plot of the studentized residuals. The window is controlled by a slide bar, which controls the value of λ. We call this a dynamic rankit plot (CW89, Section 3).

  To get the information contained in Figure 9 of CW89, first load the tree data, and then call the method:

  ```
  (load "trees")
  ```

  If you select "Dynamic Rankits" from the Graphics menu, the keywords will be set to their default values, which are different from the values used in the paper. The graph in the paper can be duplicated by typing the following command:

  ```
  (send tree-reg :dynamic-rankits :lambda '(-2 -1.5 -1 -.5 0 .33 .5 1 1.5 2))
  ```

  This method precomputes all the regressions, and may therefore take some time, especially for large data sets.

  The method as several optional keywords. The `:residuals` keyword can be used to change the specification of what to plot; the default is `:residuals 'studentized-residuals`; any other specification will use ordinary residuals in place of studentized residuals. The y-values can be shifted to plot $(Y + \mu)^{(\lambda)}$ by setting the `:shift` keyword to the number μ. The values of λ are set by the `:lambda` keyword, with default value '(-1.5 -1 -.5 -.33 -.25 0 .25 .33 .5 1 1.5). The family of transformations is set by the `:y-transform` keyword. The default is:

  ```
  :y-transform #'(lambda (c lam)
                   (let ((ans (send self :bcp c lam)))
         (if ans ans (error "Negative y"))))
  ```

  The :bcp method computes the normalized Box-Cox power family transformation with power lam and shift c. Finally the quantile function, can be changed from `#'normal-quant` to some other distribution using the `:quantile-function` keyword. An example of a specification of a quantile function is given in the discussion of the qq-plot above.

  This plot calls the method `:dynamic-plot` which can be used as the basis of other animated plotting procedures. For example, one could easily modify the code provided to give a dynamic

plot of residuals versus fitted values as a power transformation of the response or a predictor is changed.

- `:hetero-score ()` Plot and statistics for score test for heteroscedasticity. The menu item in the Graphics menu for this plot is "Non-constant Var Score". See Cook and Weisberg (1983).

  The plot is of $e_i^2/\sum(e_j^2)$ on the vertical axis versus an estimated direction on the horizontal axis. The model fit is of the form $y_i = x_i^T\beta + \epsilon_i$ with $\mathrm{var}(\epsilon_i) = \sigma^2\exp(z_i^T\gamma)$. The estimated direction is given by $z_i^T\hat{\gamma}$ where $\hat{\gamma}$ is an estimate of γ. z_i is chosen using the "Change Model" button on the plot. Another button allows toggling between squared residuals and absolute residuals. At the top of the plot, three numbers are printed. These are, respectively, the score test for nonconstant variance (essentially testing $\gamma = 0$), its asymptotic degrees of freedom, and its asymptotic p-value from a Chi-squared approximation.

- `:graphics-menu (&optional (title "Graphics"))` Adds a menu to the menu bar with the title "Graphics", or whatever alternative title you specify. This menu allows accessing all the above graphs without typing. Here is a brief description of the items in this menu. Items not described were described above.

  - "Stud. res. vs yhat" gives a 2-D plot of $\{r_i$ = studentized residuals, $\hat{y}$ = fitted values$\}$.
  - "ARES plot" gives a 2-D ARES plot.
  - "AVP – 2-D" gives a 2-D added variable plot.
  - "AVsP – 3-D" gives a 3-D added variables plot.
  - "Spin residuals, 3-D" gives a 3-D spinning residual plot. This plot has an additional plot control called "Rotations menu," described in Section 2.5.4, that can be used to get special rotations in this plot.
  - "Local influence–absolute" gives a 2-D plot of {case numbers, |direction cosines for maximum local influence for case weight perturbation|}. See Cook (1986) or Lawrance (1991) for details.
  - "Local influence–signed" gives a 2-D plot of {case numbers, direction cosines for maximum local influence for case weight perturbation}.
  - "Normal-plot" gives a normal probability plot of studentized residuals. To get a normal plot of any other quantity, the user can use the `:qq-plot` method described above.
  - "Non-constant Var. Score" sends the :hetero-score method, giving a 2-D plot.
  - "Dynamic-rankits" gives a dynamic rankit plot.
  - "Scatterplot Matrix" gives a scatterplot matrix of the predictors and the response.
  - "Other plot" allows the user to draw an arbitrary plot. When this item is selected, a dialog appears in which the users can select one or more quantities to be plotted. The choices include all the predictors and the quantity names given under the description of the plot command. If one item is chosen, a histogram is drawn; two items for a scatterplot; 3 items for a spinning plot, and more than 3 items for a scatterplot matrix. The first item chosen is put on axis 0 (horizontal or x-axis), the second item on axis 1 (vertical or y-axis), and so on.
  - "Display fit" prints the coefficient estimates and standard errors in the listener window by sending the :display message.
  - "Remove graphics menu" removes the Graphics menu from the menu bar on the Macintosh.

- `:graphs` returns, as a list, all the graphs associated with the regression model.

- `:last-graph` returns the last graph created.

2.5 Plot Controls

All of the plots produced by selecting items in the graphics menu will produce a selection of plot controls appropriate for that plot. If you type the plot message yourself, you can add a keyword, `:plot-controls nil` to remove the controls. This is likely to be desirable only for plots for publication. Many of the controls described here are available for any graph, whether created by the :plot method or not, by sending the graph the message `:plot-controls`. All plots except histograms include a symbol palette and, if the screen has color, a color palette. To color a point, select the point and then click on the color you want in the palette. This can also be done using standard XLISP-STAT menu items.

2.5.1 Histograms

Histograms have two or three plot controls, depending on the values in the plot. All controls are slide bars.

- "NumBins" is used to vary the number of equal sized bins in the plot.
- "Gauss Ker Dens" is used to fit a Gaussian kernel density estimate to the data; the slide bar controls the bandwidth. The slide bar varies from 0.01 to 0.50; to get the bandwidth used, multiply the value in the slider by the range of the visible data on the horizontal axis, so the bandwidth is determined as if the data were in the range $(-1, 1)$.
- Transformations. If the data are strictly positive, a slide bar appears controlling a power transformation for the data. The data actually displayed in the plot is given by $(y^\lambda - 1)/\lambda$, where λ is the value chosen by the slide bar (with $\lambda = 0$ corresponding to $\ln(y)$) and if $\lambda = 1$ the original data is displayed. The transformation is *not* passed back to the underlying regression model, but is used only in the display. These controls can be added to any plot by sending the :install-transform-control message. Unlike other plot controls, the transformation controls will not interact properly with other plot controls that modify the data, such as removing linear trends, or changing the underlying model by deleting a case.

2.5.2 Scatterplot Matrices

The plot controls here are slide bars for each strictly positive variable in the plot. As with histograms, these control power transformations applied to the data in the plot, but these do not update the regression model.

2.5.3 Scatterplots

A number of slide bars are given on scatter plots.

- "Rem. Lin. Trend" removes/restores a linear trend from the plot. This is done using OLS, based on all visible points.
- "OLS fit" draws the OLS line, again based on all visible points.
- "Zero line" draws a horizontal line at zero.
- "Join Points" draws a line between the points, as might be appropriate if the values on the horizontal axis are time or case numbers.
- "M-est" is a slide bar for fitting an M-estimate to the plot, using the visible data (this is fit to the plot, not to the underlying regression model), with Huber's ψ-function, and with tuning constant selected in the slide bar. The usual choice of tuning constant is about 1.25 to 1.5. A value of "NIL" on the slide bar indicates no line is fit.

- "Gauss Smooth" fits a kernel regression estimate to the visible data in the plot, with bandwidth chosen in the slide bar, and with a default Gaussian kernel.
- "Lowess" fits a lowess smoother with no robust steps, and the fraction set to the value in the slide bar. A value of "NIL" means the lowess curve is not fit.

2.5.4 Spinning Plots

Standard 3-D plots have a number of button controls.

- "Rem. Lin. Trend" removes/restores a linear trend in the plot by OLS using the visible points in the plot.
- "Orth. Axes" replaces the out-of-page axis by the making it orthogonal to the quantity on the horizontal axis. The subspace spanned by these two vectors is unchanged, but Cook and Weisberg (1989b) show that orthogonalizing will maximize the resolution in the plot.
- "Fixed scaling" toggles the scaling in the plot. When not selected the plot is in *abc* scaling, with all axes scaled to maximize resolution in the plot; when selected, the plot is in *aaa* scaling with all three axes scaled the same, which is appropriate if relative magnitudes on the three axes are of interest.
- "Extract Horiz" will create a list of data with name chosen via a dialog box of the data currently plotted on the horizontal axis on the screen. For the scaling of this list, see the next item.
- "Extract 2-D" creates a 2-D plot of the current horizontal and vertical axes. The plot controls for 2-D plots can then be used on this 2-D plot. The plot so created will not be updated when the model is changed, but it will be linked and will inherit colors, symbols, and point states. Since this plot is extracted from a 3-D plot, the coordinates on each axis are contained in the range [-1, 1].
- "Rock Plot" rocks the plot.
- "Rotations Menu" This appears only on a spinning residual plot. When selected, a menu pops up giving a number of options. "ARES rotation" gives the rotation that is equivalent to the ARES plot. "Rotate to yhat" puts the linear combination of the horizontal and out of page axes that is equivalent to $\hat{y}$ on the horizontal axis. "Rotate to e1" puts the submodel residuals on the vertical axis and submodel fitted values on the horizontal axis. "Print Screen Coordinates" prints the current value of the transformation matirx. The first column of this matrix, for example, gives the linear combination of the data that currently appears on the horizontal axis. The next three items can be used to move specific axes to specific places.

2.6 Non-graphical Methods

Here is a list of some new methods that the user may want to use. Several other internal methods that the developer might find useful are given in the code.

2.6.1 Accessor Functions that allow updating

Each of the following return slot values, or reset them if an argument is given:

- `:pweights (&optional new)` In regression-model-proto, this is the same as :weights. In glim-proto, it sets/returns the prior weights. It has been added to regression-model-proto for consistency with glim-proto.

- `:deviance (&optional new)` Returns the residual sum of squares. This has also been added for consistency with glim-proto.
- `:yvar (&optional new)` Returns the response values. For consistency with glim-proto.

2.6.2 Accessor Functions without updating

. Each of these methods returns a slot value or does a computation, but accepts no arguments.

- `:residuals-squared ()` Returns a list of $e_i^2/\sum e_j^2$. For use in heteroscedasticity plots and tests.
- `:local-inf ()` Computes the direction of maximum curvature in the likelihood displacement for the coefficient vector when case weights are perturbed and the statistic C_{max}; see Cook (1986). Returns a list whose first element is C_{max} and whose second element is the direction cosine vector.

2.6.3 Other functions.

- `:lin-combination (x)` returns a list of 2 elements, $x^T\hat{\beta}$ and its standard error, $\hat{\sigma}\sqrt{x^T(X^TX)^{-1}x}$.
- `:toggle-cases (&optional (idnum nil))` changes the 'included status of the cases in the idnum list. idnum may be a number or a list. If the list is left off or set to nil, all cases are restored to the data.
- `:quantiles (keywords)` Returns the normal quantiles of the response. There are two keyword arguments. The quantity for which quantiles are computed is set with the :x keyword. The function used to compute the quantiles is set with the :quantile-function keyword. Cases for which the 'included slot is nil are not used in computing quantiles. This is generally used internally by the probability plotting methods.
- `:bcp (c p)` computes the normalized Box-Cox power transformation of the response, adding constant c, with power p.

3 How it Works

The main features of these enhancements are (1) the close linking of the regression model and the graphs; (2) the plot controls and (3) the ability to delete cases from a fit using the plot menu. Here is an outline of how it all works. All of the graphical methods described above call the `:make-plot` message. On the first call to :make-plot, the `:make-first-plot` message is sent. This method defines a few slot values, and creates overrides for the standard `:compute` method. In particular, a slot value called 'graphs is created, and every new graph that is a plot of the data is then listed in this slot. Whenever the new :compute method is called, first the fit is recomputed, and then, if the slot 'graphs is not nil, all the graphs in this list are sent the :update message. Each plot has slots containing functions that recompute the values on its axes, and these are funcalled to get the new values for the plot. The actual updating of the plot is done via a call to the graph-proto method `:draw-next-frame`.

There are a variety of buttons and sliders that appear on various plots. These are all descendants of the graph-control-proto of Tierney (1990), and are organized in three files, overlay1.lsp, overlay2.lsp, and overlay3.lsp. The actions associated with the buttons also get properly updated when a plot is changed through the :draw-next-frame method. Each time this method is called, it first funcalls all the functions stored in a slot called 'start-next-frame, then updates the plot, then funcalls the functions in 'finish-next-frame. A typical function in 'start-next-frame might be

```
#'(lambda () (send plot :clear-lines))
```

which will remove any smoothers, or other lines on the plot. The plot is then updated, and a typical function in 'finish-next-frame will redraw any smoothers or other lines if appropriate. The order in which the functions in these lists is executed can be important. For example, if a plot is to be detrended, the trend should be removed before any smoothers are added to a plot.

The most common change to the model will be deletion/restoration of cases. This is done from the graph using two menu items. For example, if the user selects a few points from the graph, and then selects the "Delete Selection" item from the menu, the underlying regression model is sent the `:toggle-cases` message with the indices of the selected cases. The new `:compute` method is then called, and the model and plots are automatically updated.

Adding other plots and/or controls to plots should be relatively straightforward by modifying the methods provided. However, plots that are not just of data, for example, plots of log-likelihood contours, may require somewhat different treatment, including a more complex linking strategy than that one used in this code.

The code is organized into several files, so that each file is less than 32K in length (the maximum size for use with the built-in XLISP-STAT editor on the Macintosh). The regression proto methods are in reg1.lsp and reg2.lsp. The overlays (plot controls) are in overlay1.lsp, overlay2.lsp, and overlay3.lsp.

4 Graph-proto methods

A few additional graph-proto methods are included with this distribution, and these are in rgraph1.lsp. Most of the added graph proto methods will never be accessed by the end user directly, but may be of interest in developing code. However, the plot controls available in the regression plots are in fact available to ANY plot by sending the graph named, for example, `plot`, the message

```
(send plot :plot-controls)
```

In addition, the message:

```
(send plot :install-transform-control)
```

will install sliders to transform the data on each axis in a plot via a power transformation in the range (-1, 2) whenever the data is strictly positive. These controls act only on the graph, and do not cause an underlying regression model to be updated. Of course, they could be modified by a change in their :do-action method, to update a regression model as well.

5 Project-proto

A prototype called `project-proto` is also provided with this distribution. The code is in file fun1.lsp. Project-proto is useful in dynamic graphics where projections provide a useful paradigm for updating plots. The prototype is very similar to regression model proto, and its use should be clear from the code.

6 Functions

Here is a list of functions provided in fun1.lsp. Arguments for these functions are described with the code.

- `sweep` Equivalent to the sweep function in *S*.
- `rmel` removes an element from a list.
- `substr` Returns part of a string. Useful in labeling spinning plots or plot controls, where long labels can be a problem.

- `mytoggle-item-proto` An alternative strategy for specifying menu items with differing text depending on a test.
- `select-list-dialog` A dialog for selecting an ordered list of items.
- `range` Returns the range of a list.
- `midrange` Returns the midrange of a list.
- `geometric-mean` Returns the geometric-mean of a list.
- `cosangle` computes the cosine of the angle between a vector and a subspace.
- `cosangle1` computes the cosine of the angle between two vectors.
- `angle` computes the angle between a vector and a subspace.
- `hnorm-quant` computes quantiles of the standard half-normal distribution.
- `boxplot` Overrides for all the standard boxplot methods so that the boxplots produced are equivalent to the "old" boxplot command in *S*. In particular, cases with values outside the "outer fences" are shown explicitly in the graph. The points in the plot are numbered as if all the data in the plot formed a single list.
- `contour-plot` Calls the standard contour-function routine, but adds some useful mouse modes.

7 References

Atkinson, A. C. (1985). *Plots, Transformations and Regression.* Oxford: Oxford University Press.

Cook, R. D. (1986). Assessment of local influence (with discussion). *Journal of the Royal Statistical Society, Ser. B*, 133-155.

Cook, R. D. and Weisberg, S. (1983). Diagnostics for heteroscedasticity in regression. *Biometrika*, 70, 1-10.

Cook, R. D. and Weisberg, S. (1989a). Regression diagnostics with dynamic graphics (with discussion). *Technometrics*, 31,277-309.

Cook, R. D. and Weisberg, S. (1989b). Three dimensional residual plots. In Berk, K. and Malone, L., eds., *Computing Science and Statistics: Interface '89*, Washington, D. C.: American Statistical Association, 162-166.

Cook, R. D. and Weisberg, S. (1990). Confidence curves for nonlinear regression. *Journal of the American Statistical Association*, 32, 544-551.

Cook, R. D. and Weisberg, S. (1991). Added variable plots in linear regression. In Stahel, W. and Weisberg, S. (eds), *Directions in Robust Statistics and Diagnostics: Part I.* New York: Springer, p. 47-60.

Lawrance, A. J. (1991). Local and deletion influence. In Stahel, W. and Weisberg, S. (eds), *Directions in Robust Statistics and Diagnsotics: Part I.* New York: Springer, p. 141-157.

Tierney, L. (1990). *Lisp-Stat.* New York: Wiley.