12-1996

Statistical Graphics: Applications to the R and GR Methods in Linear Models

Mei Huang Wang
Western Michigan University

Follow this and additional works at: http://scholarworks.wmich.edu/dissertations

Part of the Statistics and Probability Commons

Recommended Citation
http://scholarworks.wmich.edu/dissertations/1727
STATISTICAL GRAPHICS: APPLICATIONS TO THE R
AND GR METHODS IN LINEAR MODELS

by

Mei Huang Wang

A Dissertation
Submitted to the
Faculty of The Graduate College
in partial fulfillment of the
requirements for the
Degree of Doctor of Philosophy
Department of Mathematics and Statistics

Western Michigan University
Kalamazoo, Michigan
December 1996
Statistical graphics combine statistics and graphics together to visually display analytical statistical results. They present information in a way that is very easy to understand and, furthermore, they frequently reveal information that is hidden from numerical summaries. Our purpose of this study is to explore the field of statistical graphics by investigating their application to statistical methods in the context of the R and GR methods in linear models. We consider the problems of score selection, R-diagnostics, and comparison of the R- and GR-fits.

We have developed a user friendly program which uses dynamic graphics and a user interface to perform the above problems. This program displays statistical results not only dynamically but also accumulatively. We have also developed a new graphical method called the QQBP which is an enhancement of two existing methods, the qq-plot (Wilk and Gnanadesikan, 1968) and the boxplot (Tukey, 1977). It is a very useful tool in differentiating between two distributions. Implementation of our applications is done by XLISP-STAT (Tierney, 1988) which is a statistical computer package equipped with many dynamic graphical tools. Our procedures can be easily extended to other situations in statistics.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor Dr. Joseph W. McKean for his continual guidance and support throughout this research. I would also like to thank the members of my committee, Dr. John A. Kapenga, Dr. Joshua Naranjo, Dr. Gerald L. Sievers, and Dr. Michael R. Stoline for their assistance. This thesis can not be done without the availability of XLISP-STAT. I thank Dr. Luke Tierney for making XLISP-STAT available to everyone.

I am indebted to my parents, Wen-Pin and Chui, for their patience and love. Thanks also extend to my husband, J. C., for his constant support and understanding.

Finally, I would like to dedicate this dissertation to our heavenly Father for His guidance and help in my life.

Mei Huang Wang
TABLE OF CONTENTS

ACKNOWLEDGMENTS ..................................................................................... ii
LIST OF TABLES ............................................................................................... vi
LIST OF FIGURES ............................................................................................ vii

CHAPTER

I. INTRODUCTION ........................................................................................ 1
   Overview ............................................................................................. 1
   Statistical Graphics ............................................................................ 2
      An Example ............................................................................... 3
   History and Development ........................................................ 5
   The R and GR Methods in Linear Models ..................................... 7
      Score Selection ............................................................................ 8
      R-Diagnostics ............................................................................ 10
      Comparison of the R- and GR-fits .......................................... 11
   Implementation ................................................................................. 13

II. METHODS OF STATISTICAL GRAPHICS ........................................ 18
   Animation .......................................................................................... 18
   Linking ............................................................................................... 21
   Rotation ............................................................................................. 22
   Scatterplot Matrix ............................................................................ 24
Table of Contents—Continued

CHAPTER

QQBP ................................................................................................. 25
QQ-plot ............................................................................................ 26
Boxplot .......................................................................................... 29
QQBP ........................................................................................ 30

III. APPLICATION TO THE R-METHODS ................................. 32

Menu of “Score-Selection” ........................................................... 34
Menu of “R-Diagnostics” and ”RDFFITS/R-EXT” .................... 37
Menu of “R-Diagnostics” ........................................................... 37
Menu of “RDFFITS and R-EXT” ........................................ 39

Examples .......................................................... 40

IV. APPLICATION TO THE COMPARISON OF
THE R- AND GR-FITS ............................................................ 47

Menu of “Comparison of the R- and GR-fits” ................... 50

Examples .......................................................... 53

Example 1 — Stars Data ......................................................... 53
Example 2 — Wood Data ........................................................ 57

Conclusion .......................................................... 60

V. CONCLUSION AND FUTURE WORK ................................ 62

Conclusion .......................................................... 62

Future Work .......................................................... 64

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Table of Contents—Continued

APPENDIX

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Our XLISP-STAT Program</td>
<td>65</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>142</td>
</tr>
</tbody>
</table>
LIST OF TABLES

1. Anscombe's Quartet

3

vi
# LIST OF FIGURES

1. Anscombe's Quartet ................................................................. 4
2. The Main Menu ........................................................................ 15
3. A Value-Dialog ........................................................................ 16
4. A Slider ................................................................................... 17
5. Three Snapshots of Animation Images, as $2m_1 = 2$ .......... 19
6. Three Snapshot of Animation Images, as $2m_2 = .8$ ............ 20
7. An Example of Linking ............................................................ 22
8. Snapshot of Rotation Images .................................................... 23
9. Snapshot of Scatterplot Matrix Image ....................................... 24
10. QQ-plots of Mismatch .............................................................. 28
11. An Example of Boxplot ............................................................ 29
12. QQBP of the Previous QQ-plot Example ................................. 31
13. Result of Dynamic Search ....................................................... 41
14. QQBP of Log-F(1.5,3.7) and Log-F(2,10) .............................. 42
15. Residual Plots of Log-F(2,6) (Full Data) ................................. 43
16. Residual Plots of Log-F(2,6) (Case 0 Deleted) ....................... 44
17. Residual Plots of Log-F(2,200) (Full Data) ............................ 45
18. Comparison Menu and TDBETAS Plot — Stars Data .......... 53
19. CFITS Plot of All Comparisons — Stars Data ..................... 54
List of Figures—Continued

20. Data Plot — Stars Data ................................................................. 55
21. CFITS Plot With Cases Removed — Stars Data ......................... 56
22. RDFFITS and R-EXT Plot — Stars Data ..................................... 57
23. Comparison Menu and TDBETAS Plot — Wood Data ............... 58
24. CFITS Plot of All Comparisons — Wood Data ......................... 59
25. Data Plot — Wood Data ................................................................. 60
26. RDFFITS and R-EXT Plot — Wood Data .................................... 61
CHAPTER I

INTRODUCTION

Overview

We live in an information age and often are flooded with data. An effective way to summarize and present information is thus needed. Statistical graphics combine statistics and graphics together to visually display analytical statistical results. They present information in a way that is very easy to understand and, furthermore, they frequently reveal information that is hidden from numerical summaries. The latter is especially true when they are used with modern dynamic graphics methods.

Our purpose of this study is to explore the field of statistical graphics by investigating their application to statistical methods in the context of the R and GR methods in linear models. The areas of the R and GR methods that we focus on are score selection, R-diagnostics, and comparison of the R- and GR-fits. We have developed an user friendly program which uses statistical graphics, especially dynamic graphics, and a user interface to perform the above tasks. Our program links to outside programs to obtain numerical values of various statistics and then uses statistical graphical methods to display results dynamically and
accumulatively; that is, we can show numeric values of statistics instantly and can display the history of the process. These features make data analysis fairly convenient to perform. We also developed a new graphical method called the QQBP which is an enhancement of two existing methods, the qq-plot (QQ) (Wilk and Gnanadesikan, 1968) and boxplot (BP) (Tukey, 1977). It is a useful tool in differentiating between two distributions. As we will show, it is a worthwhile exploratory procedure.

The complexity of our application lies in its implementation. It requires not only the knowledge of both fields but also of computer packages which have the ability to perform the application. Our application uses the free package called XLISP-STAT (Tierney, 1988) which is equipped with many dynamic graphical techniques. We used this package to display graphical information for robust procedures in linear models. From this application, we have learned that with proper care in the design and display, statistical graphics has unlimited usage.

Statistical Graphics

Statistical graphics are visual displays of statistical information. They can help us improve presentation, clarify statistical analyses, and reveal unknowns. They also can express concepts which are not easy to state in common language. Indeed, as the old truism says, “one picture is worth a thousand words”. High quality statistical graphics are not only pleasing to the eye but also display im-
portant statistical information. Although graphics can be an eminently potent tool to communicate ideas and information, they need to be prepared with care. Otherwise poorly constructed graphics can easily induce confusion and mislead readers (Huff, 1954). Many design principles of graphics are available. See, for example, Chambers et al. (1983), Cleveland (1985), Tufte (1983), Wainer (1984), Schmid and Schmid (1979), Cox (1978), and Tukey (1990b).

An Example

Table 1 lists some data and statistics. The first four columns are the famous data sets given by Anscombe (1973), called the Anscombe's quartet, and the last column gives their identical linear regression statistics. From this table,
we can tell that these four data sets are different yet their comparable linear regression statistics are the same. By looking at each data set individually, it is hard for us to see the relationship between x and y.

Figure 1 shows the graphical version of the Anscombe's data. The four data sets are presented by the four scatterplots of the same scales with their identical statistics plotted as the fitted line of the data. From these plot, we can tell quite easily that the Anscombe's quartet have very different data patterns and the same fitted line. Their data patterns are a random scatter around the fitted line, a curve, a straight line except for an outlier in y-space, and a vertical line.
except an outlier in x-space, respectively.

This example well illustrates the usefulness and importance of statistical graphics. These data sets, also, remind us about the importance of regression diagnostics because they can easily identify those pattern differences. Diagnostics and graphics go "hand-in-hand", as we will discuss in Chapters III and IV.

History and Development

The usage of statistical graphics can be traced back to around 1800 (Tufte, 1983). It was mainly used in official and governmental publications. Up until 1930's, graphical methods were an important part of statisticians' interest. Later the interest declined because graphical methodology was considered less rigorous compared to mathematical statistics (Feinberg, 1979). In the early 1960's, interest in the use of statistical graphics began to emerge again. This is mainly due to contributions of John Tukey and computer technology.

Tukey made statistical graphics respectable through his innovation which combines statistical theory and graphics together and offered a new insight in exploratory data analysis (EDA, Tukey 1977). His inspiration led to considerable research in this area. At the same time, advancement in computer technology not only made the production of graphics much easier and more precise but also made graphics dynamic. That is, graphical display on computer screen can be specified and changed nearly instantaneously. The user can interact with the displayed
image directly. This dynamic characteristic not only increased enormously the power of traditional static statistical graphic methods to convey information about the data but also brought great potential and created a new area of research. Currently this area of statistical, computational graphics is very active in research (Cleveland, 1987).

Current active research areas in statistical graphics are methodology, computing, and graphical perception (Cleveland, 1987). Many major events have recently occurred in this field, including new technical section, journal, and computer packages. In 1987, a new technical section of the American Statistical Association (ASA) called Statistical Graphics was established and it has grown to one of the largest sections of the ASA. A new journal called *Journal of Computational and Graphical Statistics* was introduced in 1992 with the purpose to improve and extend the use of computational and graphical methods in statistics and data analysis. New statistical computer packages dedicated in these areas are S (Becker, Chambers, and Wilks, 1984), S-PLUS (1990), XLISP-STAT (Tierney, 1988), R (Ihaka and Gentleman, 1996), Data Desk (Velleman and Velleman, 1988), R-code (Cook and Weisberg, 1994), and ViSta, *The Visual Statistics System* (F.W. Young, 1992). Also many existing packages are putting resources into graphical displays, such as SAS.
The R and GR Methods in Linear Models

As this thesis is concerned with applying statistical graphics to R and GR methods in linear models, we next introduce the linear model. Consider the linear model

\[ Y = \alpha 1 + X_c \beta + e \]

where \( Y \) denotes an \( n \times 1 \) observation vector, \( \alpha \) is the scalar intercept parameter, \( 1 \) is an \( n \times 1 \) column vector of ones, \( X_c \) is an \( n \times p \) centered design matrix having full column rank, \( \beta \) is a \( p \times 1 \) vector of regression parameters, and \( e \) is an \( n \times 1 \) vector of random errors whose components are independent, identically distributed (iid) with an unspecified distribution function \( F \), and density function \( f \).

The least square method is the traditional way to handle linear model problems, but it is highly sensitive to outliers. Recently robust methods for linear models have been developed; see Huber (1981) and Hettmansperger (1984). These methods perform similar to least squares when the data are "good" but are much more efficient than least squares for data containing outliers. They also can accommodate different shapes of error distributions and require only mild assumptions for validity. Furthermore, they are easy to grasp, quick to perform, and widely applicable. Discussions of these methods can be found in Hettmansperger and McKean (1977) as well as Naranjo and Hettmansperger (1994). In
this thesis, we will only focus on statistical graphic procedures in areas of score selection, R-diagnostics, and comparison of the R- and GR-fits.

Score Selection

The R-estimate of $\beta$, $\hat{\beta}_R$, was proposed by Jaeckel (1972) as a value which minimizes the dispersion function

$$D(\beta) = \sum_{i=1}^{n} a(R(y_i - x'_{ei} \mathbf{\beta}))(y_i - x'_{ei} \mathbf{\beta}),$$

where $x'_{ei}$ is the $i^{th}$ row of $X_e$, $R(u_i)$ denotes the rank of $u_i$, and $a(i)$ is a set of scores generated as $a(i) = \varphi(\frac{1}{n+1})$. The score (generating) function, $\varphi$, is an increasing and bounded function defined on the interval (0,1). For example, the Wilcoxon scores are generated by $\varphi(u) = \sqrt{12}(u - \frac{1}{2})$.

Hajek and Sidak (1967) showed that if

$$\varphi(u) = \varphi_f(u) = \frac{-\partial}{\partial x} \log f(x)|_{x=F^{-1}(u)} = \frac{-f'(F^{-1}(u))}{f(F^{-1}(u))},$$

then the resulting R-estimate is asymptotically efficient. Therefore, to have an efficient R-estimate, we need to choose a $\varphi$ such that $\varphi(u) = \varphi_f(u)$. In practice, however, $f(x)$ and $\varphi_f(x)$, hence, $\varphi_f$ are not known. Several methods are proposed to estimate the score function, for example, Beran (1974), Dionne (1981), Hajek and Sidak (1967), Kapenga and McKean (1989), and McKean and Sievers...
(1989). Since our focus is on the application of statistical graphics, we chose to implement the one proposed by McKean and Sievers (1989). Their selection of a score function is dependent on how much we know about the underlying density. In this thesis, we will implement this score selection procedure for survival-type data which satisfies the linear model assumption with errors being in the log-F (Kalbfleisch and Prentice, 1980) family of distributions.

A random variable \( Z \) is said to have a log-F distribution with parameter \((2m_1, 2m_2)\), if \( Z \) is defined as \( \log(X) \), where \( X \) is \( F \) distributed with \((2m_1, 2m_2)\) degrees of freedom. A distribution in this family is symmetric, positively skewed (right-skewed), negatively skewed (left-skewed), lightly tailed, and heavily tailed when \( m_1 = m_2 \), \( m_1 > m_2 \), \( m_1 < m_2 \), \( m_1 > 1 \) and \( m_2 > 1 \), and \( m_1 < 1 \) and \( m_2 < 1 \), respectively (Kalbfleisch and Prentice, 1980 and McKean and Sievers, 1989). Thus the log-F family offers a wide variety of shapes and tail weight. It is a popular family of distributions for survival type data; see Kalbfleisch and Prentice (1980). The optimal score of log-F distributions is

\[
\varphi_{m_1, m_2}(u) = \frac{m_1 m_2 (e^w - 1)}{m_2 + m_1 e^w}, \quad \text{where} \quad w = F^{-1}(u), \quad \text{and} \quad 0 < u < 1,
\]

which is bounded below by \(-m_1\) and bounded above by \(m_2\).

Since the R-estimates are consistent for any score function, residuals based on the R-method should resemble the true errors. Hence, McKean and Sievers' score selection procedure used Wilcoxon scores as an initial fit, followed by a
validity check of residual analysis. The principal tool that they used for this diagnostic analysis was the qq-plot. The ordered residuals from the initial fit form the coordinates along the ordinate axis and the population quantiles from a selected distribution form the coordinates along the abscissa axis. Linearity of the qq-plot indicates the appropriateness of the selected distribution and hence scores. Departures from linearity often indicates what scores to pursue next.

**R-Diagnostics**

The assessment of model fitting is very important in the analysis of linear models as seen in the Anscombe's example. Tools of the assessment include residual plots and diagnostic techniques. They can help us detect curvature not accounted for by the fitted model, data points at which the model and the data differ greatly (such points are called outliers), and data points which have a great influence on the fit of the model (such points are called influential points). References of this area can be found in Cook and Weisberg (1982), Belsley, Kuh and Welsch (1980), Hocking (1985), Chatterjee and Hadi (1986), and McKean, Sheather, and Hettmansperger (1990, 1991).

Suppose that the proposed model is correct, McKean, Sheather, and Hettmansperger (1990, 1991) showed that residuals from the R-fit can be used in the same way as their least square counterparts. The interpretation of residual plots from R-fit is the same as in the least square method. They also developed re-
gression diagnostics for R-estimates, such as the R internal t-statistic (R-INT), R external t-statistic (R-EXT), and Rdffits. Use of these regression diagnostics is the same as in least square method. The R-INT and R-EXT flags outlying cases while the Rdffits measures the impact of an individual case on the R-fit. Benchmarks used to flag cases are $\pm 2$, $\pm 2$, and $\pm 2\sqrt{p/n}$ for R-INT, R-EXT, and Rdffits, respectively. Computation of R-estimates, R-residuals, and R-diagnostics can be obtained from the package RGLM (Robust General Linear Model) which is implemented by Kapenga, McKean and Vidmar (1988).

Comparison of the R- and GR-fits

The R- and GR-estimator can give quite different fits for a data set. Let us recall the definition of both estimators. The R-estimator was proposed by Jaeckel (1972) and defined as

$$\hat{\beta}_R = \min_{\beta} D(\beta) = \min_{\beta} \sum_{i=1}^{n} a(R(y_i - x'_{c,i}\beta))(y_i - x'_{c,i}\beta),$$

where $D(\beta)$ is Jaeckel's dispersion function. The GR-method was introduced by Sievers (1983) and generalized by Naranjo and Hettmansperger (1994). The GR-estimate of the regression parameter, $\hat{\beta}_{GR}$, is defined as the estimate which minimizes the Sievers' weighted dispersion function, $D_w(\beta)$, that is,
\[ \hat{\beta}_{GR} = \min_{\beta} D_w(\beta) = \min_{\beta} \sum_{i<j} b_{ij} |(y_i - x_{c,i}^t \beta) - (y_j - x_{c,j}^t \beta)| \]

where \( b_{ij} \) are weights. If the weights are identically equal to one, then the GR-estimate is the Wilcoxon R-estimate. In this thesis, we will use the Mallow-type weighting scheme \( b_{ij} = w_i w_j \), and

\[
w_i = \min \left( 1, \left( \frac{c}{(x_i - \mu)^t S^{-1} (x_i - \mu)} \right)^{k/2} \right),
\]

where \( \mu \) and \( S \) are the minimum volume ellipsoid (MVE) measures of location and scatter (Rousseeuw and van Zomeren, 1990). For the computations in this thesis, we set the cutoff value \( c \) at the 95th percentile of \( \chi^2(p) \). Note that the severity of downweighting increases with \( k \), the GR-exponent. If \( k=0 \) the resulting estimator is the Wilcoxon R-estimator.

The R-estimator is highly efficient but has breakdown 0, while the GR-estimator is not as efficient but has high breakdown. Also the R-methods are robust to outlying y-axis values but not the outlying x-axis while the GR-methods are robust to both outlying x-axis and y-axis. The R-estimators have the ability to detect and fit curvature, while the GR-estimators are hampered in this situation; see Cook, Hawkins and Weisberg (1992), McKean, Sheather and Hettmansperger (1993, 1994), and Naranjo et al. (1994). Comparing these two fits can reveal useful diagnostic information. McKean, Naranjo, and Sheather (1996a,b) proposed...
statistics $TDBETAS_r$ to measure an overall difference between the two fits and $CFITS_{R,i}$ to identify points on which the two fits differ greatly.

The statistics $TDBETAS_r$ is defined as

$$TDBETAS_r = (\hat{\theta}_R^* - \hat{\theta}_{GR}^*)' A_R^{-1}(\hat{\theta}_R^* - \hat{\theta}_{GR}^*)$$

where $\hat{\theta}_R^* = (\hat{\alpha}_R, \hat{\beta}_R)'$ and $\hat{\theta}_{GR}^* = (\hat{\alpha}_{GR}, \hat{\beta}_{GR})'$ are the parameter estimates including intercept, and $A_R = \text{Cov}(\hat{\theta}_R^*)$. To see the difference in fitting is significant or not, a benchmark of $4(p + 1)^2/n$ is used; see McKean et al. (1996). We will flag $TDBETAS_r$ as large whenever it exceeds its benchmark. When $TDBETAS_r$ is large, we will use

$$CFITS_{R,i} = \frac{\hat{Y}_{R,i} - \hat{Y}_{GR,i}}{SE(\hat{Y}_{R,i})}$$

to investigate differences in the fits of individual cases. Cases with value of $CFITS_{R,i}$ exceeding its benchmark $2\sqrt{(p + 1)/n}$ are flagged. They are the points where the R- and GR-fits differ greatly and they should be investigated, because they often largely determine the outcome of the analysis or the direction that further analyses should take.

Implementation

There are many software packages with dynamic graphics capacity available, such as XLISP-STAT (XLS) (Tierney, 1988), S-Plus (1990), DataDesk (Velle-
man and Velleman, 1988), SPIDER (Haslett, et al., 1990), ISP (1988), and SAS-INSIGHT (1991). We have chosen XLS as our means of implementation because it is free, flexible, and it, especially, contains many tools for dynamic graphics as well as the user interface. We used version 2.1 release 3.44 of XLS on UNIX workstations (SUN-Sparc station 20) with the X11 window system.

XLISP-STAT is a free, functional, extendible, and object-oriented environment for statistical computing and dynamic graphics. It is an interactive system based on XLISP (Betz, 1989), a dialect of LISP language, and influenced strongly by S (Becker, et al., 1984). XLISP-STAT provides many building blocks of dynamic graphics techniques. It allows statisticians without specialized training in graphics programming to experiment with and develop new dynamic graphics techniques. In addition to the statistical computing and dynamic graphics facilities, XLS also provides many user friendly graphical interfaces such as the menu and dialog interfaces. These tools really make programming flow much easier. XLISP-STAT can be run on Apple Macintosh, IBM PC and UNIX machines and it is equipped with FORTRAN and C interfaces.

We use XLS to develop a user friendly program which is organized in terms of menus and lists in the Appendix A. The main menu is displayed on Figure 2. We use the submenu “R-estimates” to generate log-F(2m₁, 2m₂) scores and run the RGLM program to obtain R-estimates. To search a proper set of log-F (2m₁, 2m₂) scores for a given set of studentized residuals, we will choose the submenu “Score
Figure 2. The Main Menu.

Selection”. Results from various RGLM runs are displayed in many styles in submenu “R-Diagnostics”. We can obtain statistics of RDFFITS as well as R-EXT and delete observations in submenu “RDFFITS/R-EXT”. To compare results from a R-fit to various GR-fits, we will click the submenu “Comparison of the R- and GR-fits”.

Some user interface tools are used in our implementation. They are the message-dialog, the value-dialog, the menu-dialog, and the slider. Since we use these tools throughout the thesis, we will introduce them here. A message-dialog displays an message on a window with an OK button. After we have read the message, we click the OK button. A value-dialog is used to obtain information from the user and it looks like the one on Figure 3. It contains some editable value fields with default values, an OK button, and a Cancel button. To change the default value on a field, we need to locate the cursor to the field and edit it,
then followed by a click of the OK button. When we click the OK button, the current contents of the editable value field are used. If we click the Cancel button, then the default values are used. A menu-dialog lists items of choice as seen in Figure 2 for our main menu. When an item is chosen, the button in front of this item will be highlighted. We choose an item by clicking the button in front of the item and the OK button. Clicking the Cancel button will do nothing except to refresh the menu.

A slider is a dialog with scroll bars and value display fields as seen in Figure 4. When the scroll bar is moved, the displayed value is changed and an action related to the new value is taken. The scroll bar can be moved either gradually or quickly. To change the value in a scroll bar gradually to the left (or right) of the current position, we just click the left (or right) arrow button at the left (or right) end of a scroll bar. To change values in a scroll bar quickly, we first
locate the cursor to the inside of a scroll bar. A sign with two arrows will appear when this is done. Then click the left (or right) button of a mouse to move value in a scroll bar to the left (or right) of current value quickly. Click the middle button of the mouse will bring the value on a scroll bar to the position of the cursor.

Figure 4. A Slider.
CHAPTER II

METHODS OF STATISTICAL GRAPHICS

Visualization of graphics implies a process of information decoding (Cleveland, 1993). Methods of statistical graphics play a crucial role in this process. In this section we will introduce those methods used in this dissertation — animation or dynamic parameter control plot (Tukey, 1973; Becker, Cleveland, and Wilks, 1987; Cook and Weisberg, 1989), rotation (Fisherkeller, Friedman and Tukey, 1975; Becker, Cleveland, and Wilks, 1987), scatterplot matrix (Chambers et al., 1983), and linking (Diaconis and Freidman, 1983; McDonald, 1983; Becker, Cleveland and Wilks, 1987). We will also discuss the QQBP method.

Animation

The common definition of the term animation is that animation makes pictures into a movie. Similarly, the dynamic graphical method in statistical graphics, animation, produces similar movie-like effects. It creates a series of images of a graphical object by varying some parameter of the object. As value of the parameter changes, the updated image of the object will be shown on computer screen instantaneously. Implementation of this method can be done by using two computer windows; one window displays the image of the graphical object and the
other controls values of the parameter. This implementation produces the movie effect of the graphical object which, hopefully, leads to insight about the object.

There are many applications of animation in statistics. For example, as shown in Figure 5, we can use it to learn the effect of $2m_1$ and $2m_2$ on the look of a qq-plot with both log-F quantiles. We generated a set of 76 quantiles from the log-F(2,2) distribution and we will use animation to plot various qq-plots, using different log-F($2m_1$, $2m_2$) quantiles. The log-F(2,2) quantiles are treated as the given data and put on the Y-axis. Quantiles of the other log-F distribution chosen from the slider, are put on the X-axis. We found that while $2m_1$ is fixed, increasing $2m_2$ will

![Figure 5. Three Snapshots of Animation Images, as $2m_1 = 2$.](image)
make the look of a qq-plot changed from approximately concave down to straight and from straight to concave up as seen on Figure 5. Similarly, when we increase value of $2m_1$ as $2m_2$ is fixed, the look of a qq-plot will move from approximately concave up to straight, and from straight to concave down as seen on Figure 6. This finding is consistent with our knowledge about the log-F distribution. That is, the weight (density) of the distribution will be shifted to the right (left) when we increase value of $2m_2$ ($2m_1$) as $2m_1$ ($2m_2$) is fixed.

Figure 6. Three Snapshot of Animation Images, as $2m_2 = .8$.

The above example is just only one instance of application of animation. There are still many others. For example, we can use it to see the effect of
a parameter to its density function, power transformation to a probability plot (Fowlkes, 1969), and new added variable to the existed model (Cook and Weisberg, 1989).

### Linking

Linking is a graphical method which can tie two or more plots together in some way. If plots are linked, then selecting (highlighting) a point in one plot will highlight the corresponding point in all the other plots, automatically. This feature enables us to see the same observations in different setting, so that we can understand the characteristics of some points in a more detail way.

There are many ways of linkage, for example, by indices, symbols, colors, and some other common properties of the data. The most common way of linkage is by the index of observation. It goes without saying that there must be a valid reason to link plots; otherwise, their graphical information maybe misleading. Suppose, we want to study relationship of three R-diagnostic statistics, studentized residuals, leverage, and RDFFITS, of a data set. We used the R-method (using Wilcoxon scores) to fit the stack loss data (Daniel and Wood, 1971; Andrews, 1974; Hettmansperger and McKean, 1977). Results of those R-diagnostics and their benchmarks are presented by three plots of case numbers versus their values with the benchmark line, respectively. We link these three plots together by case number. From this linkage, we learned that observations behave differently
under these statistics. For example, when we highlight points beyond benchmark of studentized residuals, we got cases 0, 2, 3, and 20 marked. At the same time these cases are highlighted at the other plots as seen on Figure 7. All four cases are within benchmark of leverage, though case 2 and 20 are close to the boundary. Not all of these cases are beyond the benchmark of RDFFITS, though case 3 is at the margin. Thus linking helps us to understand the behavior of outliers.

Figure 7. An Example of Linking.

Rotation

Rotation is a way to view three-dimensional data using a two-dimensional display device — computer screen through revolution. It is done by smoothly updating a sequence of projections onto the computer screen so that the point cloud appears to rotate (Cook and Weisberg, 1989). The primary use of rotation is to provide a reasonably practical way to view three dimensions (Cook and
Weisberg, 1989). It is a puissant tool to detect relationship among three variables. This method allows us to quickly gain insight into possibly hidden relationships in three-dimensional data. For data set with more than three variables, we can use rotation to view the data three variables at a time. Below is an example to illustrate usefulness of rotation.

A sample of 61 observations with three variables, $X_1$, $X_2$, and $X_3$, is generated. The first variable $X_1$ is a sequence of numbers from -3 to 3 with increment of .1. Variable $X_2$ is a random sample from the log-$F(2,10)$ distribution. The last variable $X_3$ is a linear combination of the previous two variable as $X_3 = 2X_1 + X_2$. We use rotation to plot these data to understand relationship among them. After rotating the three-dimensional data cloud, we found that these data are on a two-dimensional plane, as seen on Figure 8.

![Figure 8. Snapshot of Rotation Images.](image-url)
A Scatterplot matrix is a graphical method to display multivariate data using proper arrangement of pairwise scatterplots in a matrix format as seen on Figure 9. For a three-dimensional data, we can imagine a box enclosing the data and the scatterplot matrix shows the views of the data from all six faces. It enables us to visually link observations and detect relationships that can not be seen from individual scatterplots. Even though this method seems very simple, it is a very useful graphical tool similar to its root graphics — scatterplot. Scatterplot is a simple way to portray relationships between two variables using the coordinate principle. Its use can be traced back to the year 1669 (Costigan-Eaves, 1984).

Although it is a very old graphical method, it is still the most popular graphical
tool for two-dimensional data.

In addition to the proper arrangement of pairwise scatterplots, we also can link, delete, label, and highlight points on all the pairwise scatterplots under the XLS implementation. There are many application for this method. One useful application is the view of conditional distribution. That is, we can view the data as the value of a variable is fixed.

Comparing rotation and scatterplot matrix, both methods can be used to investigate relationship of multi-variables. They are complement to each other; one is not superior to the other. Rotation is good to see relationship of three variables. We can view the three-dimensional data cloud at any direction to reveal any hidden relationship of the three variables. However, it is limited with the number of variables only three at a time. Although a scatterplot matrix is a good way to see pairwise relationship of all variables, it is limited with its angle of view.

QQBP

The QQBP is a new statistical graphics method which displays quantiles of two data sets jointly and marginally. It is a quantile-quantile plot (QQ), (Wilk and Gnanadesikan, 1968; D'Agostino and Stephens, 1986; Rayner and Best, 1989), with an addition of a boxplot (BP), (Tukey, 1977), of each data set in the margins of the plot. This method is an enhancement of the qq-plot and is a better
tool in investigating the differences between two distributions; for example, the
two marginal boxplots can help us to identify shape differences between the two
distributions. To understand this method better, we first discuss the qq-plot and
boxplot.

**QQ-plot**

A qq-plot is the plot of quantiles of one distribution against the corre­
spending quantiles of another distribution. Most often it uses the empirical
distribution of a data set and a theoretical distribution to see the fit of the the­
oretical distribution to the empirical distribution. In this case, the qq-plot is
often called the theoretical qq-plot or probability plot (Chambers, et al., 1983).
The most common usage of this is the check of normal assumption for residuals
from a linear model. Usually we put the ordered sample values (quantiles of the
empirical distribution) on the abscissa (Y-) axis and the the corresponding (us­
ing $p_i = F_n(x_{(i)}) = \frac{i-c}{n-2c+1}$, where $F_n$ is the empirical distribution function and
$0 \leq c \leq 1$) theoretical quantiles on the ordinate (X-) axis. The choice of $c$ is not
crucial. We will use $\frac{1}{2}$ for $n > 10$ and $\frac{3}{8}$ for $n \leq 10$ as recommended by Blom
(1958).

When the two distributions are identical, their corresponding qq-plot will
be a straight line through the origin with unit slope. If the two variables are a
linear transformation of one and other, then their corresponding qq-plot will be
a straight line. This straightness of a qq-plot indicates that the two distributions have the same shape except for, possibly, differences in location and/or scale. This linear invariance property makes the qq-plot a particularly useful tool. First, because linearity is the geometric configuration that is the easiest one for the human eye to perceive. Second, because a single theoretical qq-plot compares a set of data not just to one theoretical distribution, but simultaneously to a whole family of distributions with different locations and scales. For instance, a single normal probability plot constructed using quantiles from the standard normal is sufficient to test the data against all normal distributions.

When large or systematic departures from straightness are observed in a qq-plot, we can conclude that, apart from location and scale, the shapes of the empirical distribution and theoretical distribution do not match. In this case, the plot provides a warning that the match is poor and it also suggests the nature of the mismatch. Since the log-F family has various shapes of distributions, we will use it to generate data to illustrate the mismatch.

Seven sets of 76 quantiles representing different combination of distribution shapes are generated from the log-F family and used to show the mismatch. Figure 10 displays the qq-plot of these quantiles. They are symmetric (log-F(2,2)) vs. symmetric (log-F(.5,.5)), symmetric (log-F(2,2)) vs. left-skewed (log-F(.5,4)), left-skewed (log-F(.5,2)) vs. left-skewed (log-F(.5,1)), left-skewed (log-F(.5,2)) vs. left-skewed (log-F(.5,100)), and right-skewed (log-F(2,.5)) vs. left-skewed (log-
Figure 10. QQ-plots of Mismatch.

F(.5,1)). Note that the third and fourth plots are from same shape of distributions, but their qq-plots look quite different. Also the second and fifth plots are from different shape of distributions, but their qq-plots look quite similar. These plots show that it is hard to judge the shape of distributions from a qq-plot only.

We need some additional information in a qq-plot to help us to identify the shape of a distribution. A candidate of this is the boxplot which will be discussed at the next section.
Boxplot

A boxplot ties descriptive statistics and graphical object together to investigate the shape of a data distribution. It not only displays the location, spread, and shape of a distribution but also shows those unusual observations (outliers) in a data set. Furthermore, it is popularly used to compare different distributions. For better comparison, some variations of the boxplot, such as variable-width and/or notched boxplots, are used (McGill, Tukey, and Larsen, 1978).

Figure 11 is an example of boxplot using 30 quantiles generated from the log-F(1,30) distribution. The middle half of the data are enclosed in a rectangle box with ends at the first quartile, $Q_1$, and the third quartile, $Q_3$. Inside the box is a bar denoting the median, a measure of location. The length of this box is the interquartile range, $Q_3 - Q_1$, which is a measure of spread. Tukey adds 1.5 times the interquartile range to the end of the box as a fence of the

![Figure 11. An Example of Boxplot.](image)
data. Then adjacent values, the observations within and closest to the upper and lower fence, are plotted as the appendages to the box. The relative distances of median to quartiles and adjacent values reveal information about the shape of data distribution such as symmetry and skewness. In addition to the box and appendages to the box, observations which are outside the fence are plotted individually to bring special attention to these outside values.

**QQBP**

As mentioned earlier, the qq-plot can not help us to see shape difference between two distributions directly. To improve it, we will add boxplots of marginal data sets to the margin of the qq-plot. This addition makes a lot of sense because both the qq-plot and boxplot display quantiles. By adding these two plots together, we can see quantiles of two distributions jointly and marginally. For implementation of this new graphical method, we need to consider the location for boxplots. We choose to put them into the open margins instead of the axis for the balance look of the plot. To see the benefit of this new method, let us redo the same example of the qq-plot by replacing it with QQBP as shown on Figure 12. With boxplots at the margins, the QQBP really help us to see difference between the two distributions.
Figure 12. QQBP of the Previous QQ-plot Example.
CHAPTER III

APPLICATION TO THE R-METHODS

In this Chapter, we will discuss the application of statistical graphics to the R-methods in linear models on areas of score selection and R-diagnostics. Since both areas are diagnostics oriented, we will combine them together to give a better and more detail picture about the model fit. Our application combines the algorithms proposed by McKean and Sievers (1989) and McKean and et al (1990). Here, we will assume that the errors are from the log-F family; that is, we will limit our choice of scores to those generated from the log-F distributions. The QQBP is the main tool used in our procedure of score-selection, while the linking is used mainly for the R-diagnostics. Animation is used in both areas.

To simplify writing, we will denote the studentized residuals simply as \( \hat{r} \), since they are used often in our application. For a given data set, we will follow the process below:

1. Generate the initial log-F(2,2) (Wilcoxon) scores.

2. Run the RGLM program using the previous generated scores to obtain the R-fit results.

3. Confirm the choice of scores using QQBP of \( \hat{r} \) from the previous fit and quantiles of the log-F distribution which was used to generate scores. If
the QQBP is linear then we are satisfied with the choice of scores. Otherwise an animation of QQBP will be performed to search for \((2m_1, 2m_2)\) which will provide a reasonably linear QQBP. Then this new found value of \((2m_1, 2m_2)\) will be used to generate a new set of log-F scores which will then be utilized to run the RGLM. This searching process will be iterated until we are satisfied with the choice of scores. Results from various RGLM run will be stored and used in the next step for R-diagnostics.

4. Display results of the R-diagnostics from various RGLM runs using the linking method to see residuals from different fits. Detail R-diagnostics including RDFITS and R-EXT will be performed next for either confirmation or outlier deletion depends on the check of R-residuals is satisfactory or not.

5. Perform the detailed R-diagnostics which are useful in detecting influential points. Cases flagged by these diagnostics need to be investigated. If results from these detail R-diagnostics are satisfactory then we are done with the process. Otherwise we have the opportunity to delete some outlying or influential cases and re-run the RGLM.

6. The process will be repeated until we are satisfied.

The XLS program which implements the above process is organized in terms of menus. The main menu includes the menus of “R-Estimates”, “Score-Selection”, “R-Diagnostics”, and “RDFITS/R-EXT”. In menu “R-estimates”, we are given choices for values of \(n\), \(2m_1\), and \(2m_2\). Those values will be used
to generate the log-F($2m_1, 2m_2$) scores to run the RGLM program to obtain R-estimates. If we are not sure about what values to provide, a set of default values are offered for our convenience. The default value of $n$ is the sample size of the data. The default value for ($2m_1, 2m_2$) is (2, 2) for the initial R-fit and the chosen value from step 3 for the following R-fits. Details of menu “Score-Selection”, “R-Diagnostics”, and “Rdffits/R-Ext” as well as an example will be given in the following sections.

Menu of “Score-Selection”

If the assumption that the errors follow a log-F family is reasonable then $\hat{r}$ should resemble the true errors with some log-F($2m_1, 2m_2$) distribution, because the R-estimates are consistent for any score function. Therefore our goal is to find a proper set of ($2m_1, 2m_2$) which will generate a log-F distribution that is similar to the distribution of $\hat{r}$. In this menu, we are given a set of $\hat{r}$ and would like to either confirm the choice of scores or search for a proper value of ($2m_1, 2m_2$) to be used to update scores.

This menu begins with value-dialogs for ranges of $2m_1$ and $2m_2$. The low, middle, and high values of a range are asked in the value-dialog. They will be used to generate the range that is used in the scroll bar for choice of ($2m_1, 2m_2$) values. The range will consist of thirty-nine numbers of which twenty are generated from low to middle and middle to high separately. Here are some remarks on values of
low and middle of the range. First, to avoid the error of calculating the logarithm of a value whose machine precision is zero, we suggest that the low value be at least .5. Second, we usually put the value of \((2m_1, 2m_2)\) that is used to generate scores into the middle value of the range. Because the QQBP of the middle value is shown first. We then can confirm the choice of scores right away. A QQBP combined with a slider for the parameters is used as our search tool. The studentized residuals are plotted on the Y-axis. The value of \((2m_1, 2m_2)\) which was used to generate the scores used in the RGLM program are added to the label of Y-axis. The X-axis is quantiles of a log-F distribution with parameters that can be changed in the scroll bars. If the QQBP of \(r\) and quantiles of the generated log-F distribution is linear then we are satisfied with the choice of scores. Otherwise, we vary the combination of \(2m_1\) and \(2m_2\) on the X-axis to search for a linear QQBP.

To implement the search of a linear QQBP, two styles of search, dynamic and static, are combined to perform the graphical score selection. The dynamic search utilizes the slider for the X-axis \((2m_1, 2m_2)\) to screen through all possible values of \((2m_1, 2m_2)\). It is equipped with a menu-dialog of three actions: "add current result to the static search list", "change the search range of \((2m_1, 2m_2)\)", and "go to the static search". This menu-dialog will popup when we click either the OK or Cancel button in the slider. When we see a satisfactory result, we will use "add current result to the static search list" to save the result for further investigation which will be performed in the static search. If we are exhausted
with the choice of \((2m_1, 2m_2)\) on the slider, then we click "change the search range of \((2m_1, 2m_2)\)" to change the range of scroll bars. We choose to go to the static search when we want to inspect the current search result. Since we have the flexibility to go back and forth between dynamic and static search, we can investigate our search result at any time.

The static search is used to pin down our choice of \((2m_1, 2m_2)\) from the dynamic search. Three actions are included in this search. They are "view results on the static list", "go back to the dynamic search", and "done with the search (result= current display)". There are two styles to view results on the static search list, that is, "display all at once" and "display one at a time". The first style is good for comparison and the latter one is good for finalizing our choice. We can go back to dynamic search by clicking "go back to the dynamic search". With this flexibility to go back and forth between dynamic and static search, our goal of searching linear QQBP can be easily achieved. When we are done with the search, the value of \((2m_1, 2m_2)\) used in the current display is our search result.

We will then use the found \((2m_1, 2m_2)\) to generate a new set of scores. To re-run the RGLM use these scores, we will use the menu of "R-estimate". With the new set of \(\hat{r}\), we will then perform the previous score selection process. The searching process is finished when we find a satisfactory result.
Implementation of the diagnostics for R-methods is divided into two menus, "R-Diagnostics" and "RDFFITS/R-EXT", depending on the RGLM run time and scores choices. The RGLM run time of the first menu is short while the second one takes more time to compute. We can specify any scores for the first menu but the second menu is limited in score choices to those provided by the RGLM. The RGLM offers ten kinds of scores and they are Wilcoxon, positively skewed winsorized Wilcoxon, light tailed, negatively skewed winsorized Wilcoxon, winsorized Wilcoxon, Log Paretos-type 1, Log Paretos-type 2, signed-rank Wilcoxon, signed-rank light tailed, and signed-rank winsorized Wilcoxon. For our application, we will use the Wilcoxon (log-F(2,2)) and Log Paretos-type 2 (log-F(2m1,2)), and Log Paretos-type 1 (log-F(22m2)) scores.

Menu of "R-Diagnostics"

We use this menu to investigate residuals from various RGLM runs. This menu provides many styles of displays to help us to understand the residuals. It begins with a menu-dialog of actions "Display various RGLM results" and "Done". When we choose the first action, a menu-dialog with four choices of display is popup. Display choices are "All current residual plots", "Individual residual plot", "Boxplot of studentized residuals", and "None".
The first display choice of "All current residual plots" will give us all six residual plots. They are plots of "Fitted value vs. studentized residuals", "Fitted value vs. residuals", "Case number vs. studentized residuals", "Case number vs. leverage values", "QQBP of studentized residuals", and "Case number vs. scores". For all the above residual plots, we have the choice of highlighting points. Criteria for highlighting points are in the menu-dialog of "Choices of Outliers" which is popuped after all the residual plots are shown on the screen. There are three ways to highlight points using criteria of "studentized residuals", "leverage values" and "combine all of the above". We can use menu "Choices of Outliers" to see various outliers. If we have more than one RGLM run, then a slider with scroll bar representing number of runs will be popup next. In this case, we can see residuals from different runs one run at a time. Note that the XLS index begins with 0 instead of 1. That is, run 0 means the first run, and run 1 means the second run, etc.

When we choose the second display choice, "individual residual plot", a menu with choice of those six residual plots and "none" is popup. Clicking "none" will bring us back to the display menu. Choosing any of the residual plots will pop up a style menu with choices of "display-all" and "movie-like". The "display-all" style will display the chosen residual plot from all RGLM runs altogether for up to 6 runs in a screen. The "movie-like" style will use a scroll bar to control value of runs and show the chosen residual plot one run at a time. The third display
choice, "boxplot of studentized residuals", will give us a plot with boxplot of \( \hat{r} \) from various RGLM runs. The fourth display choice, "None", will bring us back to the beginning menu-dialog of choice for R-diagnostics. Clicking "Done" here, will bring us back to the main menu.

**Menu of "RDFFITS and R-EXT"**

This menu offers additional diagnostic statistics, RDFFITS and R-EXT, and outliers deletion. It begins with a menu-dialog which contains action of "Run RGLM to get RDFFITS and R-EXT statistics", "Display results", "Delete outliers to refit", and "Done". A message about the RGLM run time is given when we choose the first action. Scores choices of log-F(2,2), log-F(2m1,2), and log-F(2,2m2) are given when we choose to run RGLM to get RDFFITS and R-EXT. A process similar to the dynamic search in "Score-Selection" is performed when we use either log-F(2m1,2) or log-F(2,2m2) scores to help us to choose the proper value for \( 2m_1 \) or \( 2m_2 \).

Actions of "Display results" and "Delete outliers to refit" will not be performed unless we have done the "Run RGLM to get RDFFITS and R-EXT". With results of RDFFITS and R-EXT, we can use "Display results" to view eight residual plots altogether. They are the six plots used in the menu of "R-diagnostics" and two new plots of "Case number vs. RDFFITS", and "Case number vs. R-EXT". After those eight plots are shown on the screen, a menu-dialog for choice
of outliers is popped up. It is similar to the one used in the menu of "R-Diagnostic" with addition of "RDFFITS", and "R-EXT" items. Viewing the various outliers is helpful in outlier deletion.

To delete outliers or influential points, we will use "Delete outliers to refit". After choosing this action, an instruction of how to delete points and a confirmation of the deletion are provided. Actions in the menu of "RDFFITS and R-EXT" can be repeated until we are satisfied.

Examples

We will use the data set, laregr.dat, which was discussed by Nelson (1982, p.227), Lawless (1982, p.185), and McKean and Sievers (1989) to illustrate our application of statistical graphics to the R-methods. This data set consists of seventy-six breakdown times, \( T \), of an electrical insulating fluid under seven different levels of voltage stress, \( v \). A one-way layout with the response variable \( Y = \log(T) \) and treatment variable \( v \) is used.

The Plot 0 of Figure 13 shows the snapshot of QQBP using the initial log-F(2,2) score. This initial QQBP reveals that the log-F(2,2) distribution is not a good match to the distribution of \( \hat{r} \), since the qq-plot is nonlinear and concave down. Furthermore, the boxplot of \( \hat{r} \) is left-skewed, which indicates that we should use the left-skewed log-F distributions (ie. \( 2m_1 < 2m_2 \)) to match the \( \hat{r} \). We then vary combination of \( 2m_1 < 2m_2 \) to search proper scores for this \( \hat{r} \) generated by
Figure 13. Result of Dynamic Search.

log-F(2,2) scores. We begin with fixing 2\(m_1\) at its lowest value .5 and search for the best value of 2\(m_2\), then do the same thing for different values of 2\(m_1\). We found that the best results for fixing 2\(m_1\) at .5, 1, 1.5, 2, and 3 are 2\(m_2\) = .8, 2.1, 3.7, 6, and 30, respectively. These results are shown on the Figure 13 along with the initial QQBP.

The QQBP of log-F(1.5, 3.7), log-F(2,6), and log-F(3,30) are quite close and the log-F(1.5,3.7) is slightly better. We then used (1.5, 3.7) to update the
scores and re-run the RGLM. Result of this new fit is shown on Figure 14 accompanied by the one found by McKean and Sievers (log-F(2,10)) for comparison. Estimates and their corresponding standard errors of the fits using both scores are listed below their respective QQBP. Both of these results are satisfactory and the former is better with less value of scale estimate \( \hat{\tau} = 1.568 \) instead of 1.668.

\[
\hat{T} = 63.2 - 17.4v \\
(6.51) \quad (1.86)
\]

\[
\hat{T} = 64.0 - 17.7v \\
(6.93) \quad (1.98)
\]

Figure 14. QQBP of Log-F(1.5,3.7) and Log-F(2,10).

Recall that the R-method is robust and the QQBP needs only to identify crudely the proper score function to obtain good results (McKean and Sievers, 1989). Furthermore, we can only obtain the detail R-diagnostics, RDIFFITS and R-EXT, for either \( 2m_1 = 2 \) or \( 2m_2 = 2 \) using the RGLM. Therefore we decided to use the log-F(2,6) scores for the R-diagnostics. We then re-ran the RGLM using this new set of scores. Figure 15 shows all the residual plots including the regular and detail R-diagnostics with points beyond the benchmarks of studentized...
residuals, ±2, highlighted. There appear no systematic patterns in plots of "Fitted vs. Std Resid.", "Case vs. Std Resid.", and "Fitted vs. Resid." and thus the randomness assumption of errors is justified. The QQBP shown on the third plot of the first row also indicates a satisfactory fit. There are some points beyond the diagnostic benchmarks. They are the cases of (0, 19, 25, 28, 29, 36), (0, 1, 2), (0, 7), and (0, 19, 25, 28, 29, 36) using the criteria of studentized residuals, leverage values, RDFFITS, and R-EXT, respectively. Note that the case numbers here are in XLS ordering, that is, they begin with 0 instead of 1.

Since case of 0 is beyond the benchmarks of all criteria, we decided to delete it to investigate its influence on the fit. After the deletion, $\hat{\beta}_R$ and s.e.$(\hat{\beta}_R)$ changed from (63.7, -17.6) and (6.53, 1.87) to (65.8, -18.2) and (6.80, 1.94). The
detail R-diagnostics results after the deletion is shown on the Figure 16 which indicates an improvement in RDFFITS statistics. The leverage values were affected because the removed point is at the edge of factor space. In practice, we need to have good justification to delete any point.

Figure 16. Residual Plots of Log-F(2,6) (Case 0 Deleted).

As noted by Lawless (1982), engineers may suggest a Weibull distribution for the breakdown times in this problem. In terms of log-linear models, this means that the errors have an extreme-valued distribution. Kalbfleisch and Prentice (1980) pointed out that this distribution is essentially the limit of a log-F(2,2m₂) distribution as m₂ → ∞. From our score selection process, we know that the QQBP looks essentially the same when we increase the value of 2m₂, say to 200. This is confirmed by the almost identical residual plots of log-F(2,200) shown on
Figure 17. The value of $\hat{\beta}_R$ and s.e.$(\hat{\beta}_R)$ are quite close too and they changed from (63.7, -17.6) and (6.53, 1.87) to (64.1, -17.7) and (7.49, 2.14) when we change scores from log-$F(2,6)$ to log-$F(2,200)$. In addition to the similarity results of

log-$F(2,6)$ and log-$F(2,200)$, estimates of the extreme-valued distribution are not bounded in influence function. Therefore we recommand the R-estimates using log-$F(2,6)$ scores for this data set.

We conclude this section with a summary of this example. Our program is easy to use and it offers a complete analysis of R-method. The score selection process is very flexible and we have identified a set of log-$F$ scores which is better than the one proposed by McKean and Sievers. The QQBP is helpful in guiding us to find the proper scores and the animation really speeds up the search process.
This score selection process can be extended to other family of distributions. Our R-diagnostics process show results from various diagnostic statistics and from different fits. Linking enables us to study relationship among various diagnostic statistics and animation can help us compare results of fits from different scores and from point deletion. For this data set, we recommend to use R-estimates generated by the log-F(2,6) scores. They have a bounded influence function and have similar results as the Weibull distribution, which is suggested by engineers.
CHAPTER IV

APPLICATION TO THE COMPARISON OF THE R- AND GR-FITS

The R- and GR-estimator can give quite different fits for a data set. The R-estimator is highly efficient but has breakdown 0, while the GR-estimator is not as efficient but has a positive breakdown. Also the R-methods are robust to outliers in the y-space but not outliers in the x-space while the GR-methods have bounded influence in both the x-space and y-space. The R-estimators have the ability to detect and fit curvature, while the GR-estimators are hampered in this situation; see Cook, Hawkins and Weisberg (1992), McKean, Sheather and Hettmansperger (1993, 1994), and Naranjo et al. (1994). Thus the R- and GR-fits may differ at influential points in the x-space and/or points that determine curvature. These are precisely the cases of interest in an exploratory analysis of a data set. Hence, comparing these two fits can reveal useful diagnostic information.

McKean, Naranjo, and Sheather (1996a,b) proposed the statistics $TDBETAS_R$ to measure an overall difference between the two fits and $CFITS_{R,t}$ to identify points on which the two fits differ greatly. Recall that they are defined as

$$TDBETAS_R = (\hat{\beta}_R^* - \hat{\beta}_{GR}^*)'A_R^{-1}(\hat{\beta}_R^* - \hat{\beta}_{GR}^*)$$
and

\[ CFITS_{R,i} = \frac{\hat{y}_{R,i} - \hat{y}_{GR,i}}{SE(y_{R,i})}, \]

where \( A_R = \text{Cov}(\hat{\beta}_R^*) \). These diagnostics can expose hidden outliers, clusters of outliers, or underlying curvature. Often those points, which cause the discrepancy in the two fits lead to interesting discoveries. Furthermore, these points are often influential and, hence, determine the outcome of the analysis and/or the direction that further analyses should take.

For a given data set, McKean et al. (1996b) proposed a model criticism procedure based on the diagnostics \( TDBETAS_R \) and \( CFITS_{R,i} \) as follows:

1. Obtain the R- and GR-fits, and the diagnostic statistics, \( TDBETAS_R \) and \( CFITS_{R,i} \).

2. Compare value of \( TDBETAS_R \) with its benchmark, \( 4(p + 1)^2/n \).

3. If \( TDBETAS_R \) is less than its benchmark, this gives evidence that there are no serious outliers in the X-space. This in no way says that the model is adequate. But it does give some confidence in using the R-method to check adequacy of of the fit of the model. We then can use the process described in Chapter III to find a good R-estimate.

4. If \( TDBETAS_R \) exceeds its benchmark, then there is evidence that R- and GR-fits substantially differ. The casewise diagnostics, \( CFITS_{R,i} \), are next examined to determine what cases were the main contributors to this overall
difference. They place the cases with largest values into a group labeled Group B and place all the other cases in Group A. They also emphasized the use of dynamic plots such as brush plots with rotation. They said that "often by rotating and changing the axes for the various predictor variables, clusters of points are brought to one's attention; see Cook and Weisberg (1994)" (McKean et. al. 1996b, page 2584). Also they think the most important contribution of these diagnostics is the identification of cases in Group B which lead to an investigation of the subject matter for the rationale behind such clusters. The cases in Group B are the cases to "click on" in rotational plots. These are often in clusters of points and these clusters are exposed by rotation. Such findings may lead to an investigation of the subject matter to discover a rationale behind such clusters. This maybe the most important contribution of these diagnostics.

5. If the preceding analysis or subject matter indicates that the discrepancies are due to curvature, then we may not want to proceed with the next step and feel comfortable with the R-fit. Otherwise, we will re-fit the model using only the points in Group A.

6. If upon refitting, the new $TDBETAS_{R}$ is still larger than its benchmark, then we proceed through the above step 4 once more. This may lead us to an enlargement of Group B.

7. If the new $TDBETAS_{R}$ is smaller or the previous step was performed, then the R-diagnostics are run on the cases in Group B. This is performed by
adding points in Group B one at a time to Group A and then obtaining the R-diagnostics. Hence, the masking effect that a cluster of x-outliers has on the R-estimate is avoided. The most useful diagnostics here are RDFITTS and R-EXT to see if an excluded case follows the same model as Group A does.

Since we can specify number of comparisons and values of GR-exponents for the weights in the program to get $TDBETAS_R$ and $CFITS_{R,i}$, we can extend the previous process to compare a R-fit to various GR-fits. Also, we plot values of GR-exponent vs. $TDBETAS_R$ with its benchmark, $4(p + 1)^2/n$ to show the overall difference between the R- and various GR-fits. Our implementation of the above process is done in the menu “Comparison of the R- and GR-fits” of our XLS program. We will introduce it and provide examples in the following sections.

Menu of “Comparison of the R- and GR-fits”

This menu begins with a value-dialog which is used to request variable names of the data. These names will be used to label the axis of the plot of data. The default values of them are X1, X2, ..., and Y. The execution of program to obtain fits and diagnostics statistics comes next. It depends on the number of GR-exponents specified in the diagnostic program. We have a menu-dialog which consists of choices of different comparisons and accumulative plots. By the side of this menu is the plot of GR-exponent vs. $TDBETAS_R$ with its benchmark $4(p + 1)^2/n$ to help us decide which GR-fit (i.e., which exponent for the weights)
to be used for comparison with the R-fit. The default choice of comparison is the one with the highest value of GR-exponents for the weights.

After deciding on the choice of the GR-fit, the value of that $TDBETAS_R$ is compared with its benchmark as in the step 2 of the algorithm. If the $TDBETAS_R$ is less than its benchmark, then a message-dialog will display value of $TDBETAS_R$ and its benchmark. Whether $TDBETAS_R$ is larger or smaller than its benchmark, the plots of data and $CFITS_{R,i}$ with its benchmark $2\sqrt{(p+1)/n}$ are displayed. Even though the plot of $CFITS_{R,i}$ and data are not required in the algorithm, we still show them to give the user confirmation information.

Since the identification of Group B points is very important in the algorithm, we link the plot of $CFITS_{R,i}$ and data together to simplify the finding process. When there are more than two variables in the data, we display the data using both the scatterplot matrix of all variables and various rotation plots of the three variables determined by user. We can see all pairwise relationship of the data from the Scatterplot matrix and all three-dimensional relationships of the data from the rotation plot. Shown with the $CFITS_{R,i}$ and data plot are a message-dialog and reference plots. The message dialog gives instruction on how to choose points on the plot. The reference plots are duplicates of the $CFITS_{R,i}$ and data plot with all indices shown to help the user identify points. Selection of points is done by using a value-dialog to input the point indices. When the value of $TDBETAS_R$ is less than its benchmark, we should click the Cancel button of
this value-dialog to choose no point. If the value of $TDBETAS_R$ is larger than its benchmark, we then use this value-dialog to choose Group B points, as in the fourth step of the algorithm. After choosing the points, plots with those chosen points highlighted will be shown to confirm the choice of points in group B. If the current selection is not satisfactory, we can always re-select the points. This gives the user flexibility in point selection. With linking and flexibility in point selection, we can easily specify those points in Group B.

If we decided to delete those Group B points, then the diagnostics program will be re-run using only Group A points. If the new $TDBETAS_R$ is still larger than its benchmark, then we have one more chance to add points to Group B as in step 6 of the algorithm. After deleting points either by one or two times, we then run RGLM to get RDFFITS and R-EXT statistics as in the last step of the algorithm. They are obtained by adding points in Group B one at a time to the Group A points. The result is shown by the plots of case number of Group B points and their corresponding RDFFITS and R-EXT. This concludes the cycle of a particular comparison and we are brought back to the menu of comparison choices. If we are interested in another comparison, we can choose that comparison and go through the whole process as stated above. We also can review our process by viewing all the accumulative plots of $CFITS_{R,t}$, data, RDFFITS, and R-EXT by clicking the item “show current accumulative plots”.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Examples

We will use two examples, stars data and Wood data, taken from McKean, Naranjo, and Sheather (1996a,b) to illustrate our implementation.

Example 1 — Stars Data

This data set consists of the logarithm of measurements of the effective temperature at the surface of 47 stars and the logarithm of light intensity of these stars. This data set can be found on page 27 of Rousseeuw and Leroy (1987). We used log-temperature (the predictor) and log-intensity (the response) for the variable names. The plot of $TDBETAS_R$ on Figure 18 tells us that all GR-fits are significantly different from the R-fit, that is, values of all $TDBETAS_R$ are greater than the benchmark of .34.

Figure 18. Comparison Menu and TDBETAS Plot — Stars Data.
To have some idea about difference in GR-fits, we displayed all CFIT plots on Figure 19. Comparing these CFITS plots, we found that they all look similar.

Figure 19. CFITS Plot of All Comparisons — Stars Data.

and value of $CFITS_{R,i}$ moved closer to zero as value of GR-exponents decreased. Also there is a gap between cases of (6, 10, 13, 19, 29, 33) and the rest of data. They also stand out in the data plot as seen on Figure 20. In this simple case of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
one predictor, the rotation plot is the scatterplot. The four cases of 10, 19, 29, and

33 are very different from the other points, they are low in temperature and high
in intensity. Subject matter investigation reveals that these four cases are giant
stars and their size is much larger than the rest of the stars. The two cases of 6
and 13 are in the transition of two data clouds. Since there is not much difference
in all the comparisons, we then choose the default comparison (GR-exponent=2)
to illustrate our implementation.

Our goal in this example is to identify points that make the R- and GR-fit
different. We begin with the deletion of those four giant stars. The resulting
CFITS plot as seen on the first plot of Figure 21 indicates that cases 6 and 13
are still high in value of $CFITS_{R_j}$. Therefore, we decided to delete these two
cases also. After all the six cases are deleted, we have TDBETAS is .00 less than its benchmark .39. Also the resulting CFITS plot on the second plot of Figure 21 indicates that no more points make the two fits different. So we have identified all the points that make the two fits different and they are cases of 6, 10, 13, 19, 29, and 33.

To gain more information on these six cases which make the R- and GR-fits differ, we then run the RGLM to get their RDFFITS and R-EXT statistics one-at-a-time. Their results are shown on Figure 22. Values of RDFFITS for all these six cases are beyond the benchmark of .31 and almost all R-EXT values except for case of 13 are beyond the benchmark of 2. This result confirms our finding that cases of 6, 10, 13, 19, 29, and 33 do not confirm with the model that fits the remaining data points. Because of the masking effect, the would not have
been discovered by use of the diagnostics RDFFITS and R-EXT on the entire data set. This example is nice in that both the algorithm and subject matter knowledge lead to the same result.

Example 2 — Wood Data

This is a real data set (Draper and Smith, 1966) which was modified by Rousseeuw (1984) to contain four outliers. It contains five predictors and 20 observations. We used the default variable names, X1, X2, X3, X4, X5, and Y. Figure 23 shows the comparison menu and $TDBETAS_R$ plot which indicates a big jump from GR-exponent of 1 to 1.5. Also values of the $TDBETAS_R$ changed from not significant to significant. To learn the difference in GR-fits, we displayed the CFITS plot of all comparisons on Figure 24. Patterns of the CFITS plots for
GR-exponent of 2 and 1.5 are the same and of 1 and .5 are quite different. Even though the $TDBETAS_R$ is not significant for GR-exponent=1, there are still four cases of 3, 5, 7, and 18 which make the two fits different as seen on its CFITS plot. To understand these observations better, we display the data on Figure 25. Since we have six variables in the data, we used both scatterplot matrix and rotation plot to display the data. Because all of the rotation plots look similar we present only one of them in Figure 25.

To illustrate our implementation, we chose to compare the R-fit with GR-fit of exponent=2. Note that cases 3, 5, 7, and 18 not only make the two fits differ greatly, they are also form as a cluster at end of data. Observing this, we decided to delete these 4 points and re-run the diagnostic program. After removing those four cases, value of the $TDBETAS_R$ changes to 2.75 which is less
than its benchmark of 9. We then run the RGLM to get RDFFITS and R-EXT, their results are shown on Figure 26. It indicates that those four points are really not in the same model as the rest. Once again these results would not be obtained by use of the diagnostics RDFFITS and R-EXT on the entire data set.

Figure 24. CFITS Plot of All Comparisons — Wood Data.
Figure 25. Data Plot — Wood Data.

Conclusion

This program is very easy to use and flexible in comparing the R- and various GR-fits. We not only can identify points that make the various R- and GR-fits different but we also learn about the relationships of various GR-fits. As seen on the above two examples, relationship of their various GR-fits are quite different. Also, the combination of scatterplot matrix and rotation plot makes the information on data easily revealed. Furthermore, the linking of points on data plot and CFITS plot really helps us in identifying points for investigation.
Figure 26. RDFFITS and R-EXT Plot — Wood Data.
CHAPTER V

CONCLUSION AND FUTURE WORK

Conclusion

The purpose of this study was to explore the field of statistical graphics by investigating their application to the areas of score selection, R-diagnostics, and comparison of the R- and GR-fits. Statistical graphics use graphics to display statistical information that is easily perceived by people. High quality statistical graphics are not only pleasing to the eye but they can also clarify statistical analyses. With help of Tukey's innovative ideas on graphically displaying data along with advancement of computer technology, the field of statistical graphics has rapidly developed into a main area in statistics. Evidence of this development include the new statistical graphics section of the ASA in 1987, the Journal of Computational and Graphical Statistics in 1992, and many computer packages, such as S-Plus and XLISP-STAT, which provide high quality graphics.

The least squares method is the traditional way to handle linear model problems, but it is highly sensitive to outliers. Robust methods for linear models, such as R and GR, perform similar to least squares when the data are good but are much more efficient than least squares when the data contain outliers. They
also can accommodate different shapes of error distributions and require only mild assumptions for validity. They also are easy to grasp, quick to perform, and widely applicable.

We have developed an user friendly program which can display statistical results dynamically and accumulatively. Our program is written in XLISP-STAT and is linked to the outside program RGLM which obtains the R-fit of a linear model. We then use statistical graphical methods, mainly dynamic ones, to display these results. Animation makes pictures into movies and we apply it to the score selection process and R-diagnostics. It shows us the history of a fitting process, so that we can see the effect of scores and point deletion on a fit. Linking can tie two or more plots together and we apply it to the R-diagnostics and comparison of the R- and GR-fits. It facilitates the study of the relationships of various diagnostic statistics. We used combinations of rotation plots and scatterplots matrices to display data, which makes the data information easily to be revealed. With the addition of boxplot at the margin of a qq-plot, the QQBP, our program can graphically determine a "good" distribution for the data.

We have used our program to investigate many of the same data sets studied by authors of the methods and we have gotten better or similar results. Our process is easy and flexible to use, fast, and visually displays considerable information.
Future Work

We plan on extending our score selection process to other family of distributions. Besides our graphical process, we plan on developing some statistical analysis to help user judge a proper choice of scores. Furthermore we would like to study the effect of our score selection process on the R-analysis. The process of point selection in the program will be improved so a click and choose option can be used. In addition to the case number on a point label, we hope to include more information, for example, the value of probability on a QQBP. We also would like to extend our experience of statistical graphics into other field of statistics.
Appendix A

Our XLISP-STAT Program
(require "mei-proto")
(require "mei-fun")
(require "mei-meth")
(require "rglm-call")
(require "score-call")
(require "diag-call")
(require "rd-call")
(require "compare-call")

(defun mei ()
"The main menu for the program of Mei's thesis."

;;;; clean file "scores.dat" if it exists
(system "/bin/rm scores.dat")

;;;; initialize the program ;;;
(def initial-rglm 1) ;; 1=yes, 0=no
(def initial-rd 1) ;; 1=yes, 0=no
(with-open-file (f "xy.dat" :direction :input)
 (setq tmp (read f))
 (def n-g (read f))
 (def p-g (read f))
 (def p1-g (1+ p-g)))
 (def want-n n-g) ; will be updated in diag-call
 (def want-m1 2) ; will be updated in score-call
 (def want-m2 2) ; same as the above

(let* ((welcome-mesg
 (send message-dialog-proto :new (format nil
 "Welcome to Graphical World of R and GR
 Methods for Linear Models. Just a reminder, before we run this program please make sure files "xy.dat\" and \"INPUT\" are proper. Also, please NOTE that all index numbers are started from 0 instead of 1.")))
 (tmp (send welcome-mesg :modal-dialog))
 (main-menu (send choose-item-dialog-proto :new "R and GR Methods for Linear Models"
 (list "R-Estimates"
 "Score-Selection"
"R-Diagnostics"
"RDFFITS/R-EXT"
"Comparison of the R- and GR-fits"
"Exit")
:location '(400 3)
:title "Main Menu"
:initial 0
:show nil)))

(do ((choice 0)) ;; initialize/update
  ((equal choice 5) "Good Bye");exit condition
  (setf choice (send main-menu :modal-dialog))
  (when (and choice (< choice 5))
    (cond ((= choice 0) (rglm-call))
          ((= choice 1) (score-call))
          ((= choice 2) (diag-call))
          ((= choice 3) (rd-call))
          ((= choice 4) (compare-call))))))
(provide "mei,proto")

(defun mei-sequence-scroll-item-proto
  '(sequence display-sequence value-text-item) () scroll-item-proto)

(defun mei-sequence-scroll-item-proto :isnew
  (x &key text-item (size '(180 16)) location action display)
  (let* ((sequence (coerce x 'vector))
         (display (if display (coerce display 'vector) sequence)))
    (setf (slot-value 'sequence) sequence)
    (setf (slot-value 'display-sequence) display)
    (setf (slot-value 'value-text-item) text-item)
    (call-next-method :size size
                      :location location
                      :min-value 0 :max-value (1- (length sequence))
                      :page-increment 5
                      :action action)))

(defun mei-sequence-scroll-item-proto :scroll-action ()
  (send self :display-value)
  (pause 5)
  (send self :user-action))

(defun mei-sequence-scroll-item-proto :do-action ()
  (send self :display-value)
  (send self :user-action))

(defun mei-sequence-scroll-item-proto :value
  (optional (val nil set))
  (when set (call-next-method val) (send self :display-value))
  (call-next-method))

(defun mei-sequence-scroll-item-proto :display-value ()
  (if (slot-value 'value-text-item)
      (send (slot-value 'value-text-item) :text
            (format nil "" ,lf"
                      (elt (slot-value 'display-sequence)
                          (send self :value) ))))

(defun mei-sequence-scroll-item-proto :user-action ()
  (if (slot-value 'action)
(funcall (slot-value 'action)
    (elt (slot-value 'sequence) (send self :value))))
(require "mei-proto")
(provide "mei-fun")

(defun ppoints (num)
"This function calculates the ppoints for a given num. Ppoints is used in qq-plot"
(let ((i (iseq num)))
  ;; Note that value in LISP starts from 0 instead of 1
  (if (> num 10) (/ (+ i .5) num)
      (/ (+ i .625) (+ num .25)))))

(defun mei-yn-dialog (&key
  (title "To Confirm Action")
  (location (list 220 600)) (mesg "Do the action?")
  (action-yes nil) (action-no nil))
"Display a yes-no dialog. Note: the action should be in a lambda function form."
(let*
  ((mesg mesg)
   (yes (send modal-button-proto
              :new "Yes"
              :action action-yes))
   (no (send modal-button-proto
           :new "No"
           :action action-no))
   (yn-dialog (send modal-dialog-proto
                :new (list mesg (list yes no))
                :title title
                :location location)))
  (send yn-dialog :modal-dialog)))

(defun get-1-value (&key
  (label "2ml =") (value 0) (title "Get a value")
  (location (list 100 100)))
"Set up a value-dialog to get a value."
(let*
  ((lab (send text-item-proto :new label))
   (val (send edit-text-item-proto
            :new (num-to-string value) :text-length 5))
   (cancel (send modal-button-proto
                :new "Cancel" :action nil)))
(ok (send modal-button,proto
  :new "OK"
  :action #'(lambda ()
    (list (send val :text))))))
(value-dialog (send modal-dialog,proto
  :new (list (list lab val)
    (list ok cancel))
  :title title
  :location location)))
(car (string-to-number
  (send value-dialog :modal-dialog))))

;;; Be Careful with the NIL input
(defun sequence-selection
  (fekey (str in 1 1") (low .5) (middle 2) (high 4)
    (length 6) (location (list 10 470)))
"Create an input dialog to set up a sequence to
be used in sequence-scroll. This sequence will
have 40 points beginning at low, continue to
middle, and end up at high. This function will
return the resulting sequence"
(let*
  ((label-1 (send text-item,proto :new "Low="))
   (label-m (send text-item,proto :new "Middle="))
   (label-h (send text-item,proto :new "High="))
   (value-l
     (send edit-text-item,proto
      :new (num-to-string low)
      :text-length length))
   (value-m
     (send edit-text-item,proto
      :new (num-to-string middle)
      :text-length length))
   (value-h
     (send edit-text-item,proto
      :new (num-to-string high)
      :text-length length))
   (cancel (send modal-button,proto
     :new "Cancel" :action nil))
   (ok (send modal-button,proto :new "OK"
      :action #'(lambda () (list
        (send value-l :text) (send value-m :text)))
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(send value-h :text))))

(range-dialog
  (send modal-dialog,proto
    :new (list (list label-1 value-1)
              (list label-m value-m)
              (list label-h value-h)
              (list ok cancel))
    :title (concatenate 'string
                  "range of " string)
    :location location))
  (result (send range-dialog :modal-dialog)))

(when result
  (setf result-n (string-to-number result))
  (remove-duplicates (append
    (rseq (select result-n 0)
          (select result-n 1) 20)
    (rseq (select result-n 1)
          (select result-n 2) 20))))))

(defun rearrange-list (data order)
"This function will re-arrange the DATA in the order of ORDER."
  (let* ((len (length data))
         (result (repeat nil len)))
    (dotimes (i len)
      (setf (select result i)
            (select data (select order i))))
    result))

(defun match (short long)
"This function will find the position of SHORT elements in the position of LONG. eg:
(match (list 2 3) (list 5 1 0 3 2 4)) => (4 3)"

  (setf num (length short))
  (setf want (repeat nil num))
  (dotimes (i num)
    (setf tmp (select short i))
    (when tmp
      (setf (select want i) (position tmp long))))
  want)

(defun read-data-R (filename)
"This function will read in data in the format of RGLM input and return a list of list (n p) and data-array (dimension n p+1), eg:((2 20) #2A((1 2) (3 4))).
To get COLUMNS of data-array as a list of lists, we use

(array-to-nested-list (transpose data-array))"

(with-open-file (f filename :direction :input)
(let*
  ((stars (read f))
   (n (read f))
   (p (read f))
   (p1 (+ p 1))
   (data (make-array (list n p1)))
   (dotimes (i n)
     (dotimes (j p1)
       (setf (select data i j) (read f))))
   (list (list n p) data)))

(defun write-data-R (data filename)
"This function will write DATA into a file
FILENAME with RGLM format, where
DATA is a list of list and array."

(let* (np (select data 0))
  (n (select np 0))
  (p (select np 1))
  (xy (select data 1))
  (dim (array-dimensions xy))
  (nrow (select dim 0))
  (ncol (select dim 1)))
(with-open-file (f filename :direction :output)
  (format f "*****"%n)
  (format f "%d %d" n p)
  (dotimes (i nrow)
    (dotimes (j ncol) (format f "-%f" (aref xy i j))))
  (format f "-%%"))))

(defun run-rglm ()
"Run RGLM with data xy.dat and scores scores.dat"

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(setq f (open "RGLM.IN" :direction :output))
(format f
"title \"trial\"-\%filexy xy.dat-\%rzes-\%score useres scores.dat-\%")
(close f)
(system "/home/faculty/rglm/rglm/bin/rglm-f")

(defun rglm-rdffits ()
"This function will run RGLM to have statistics for RDFFITS etc. It returns 1 for to run, or 0 for NOT to run."
(let*
((mesg (format nil "Do you really want to obtain RDFITTS statistics?%(NOTE: This will take a while, please be PATIENT)")))
(prompt (send text-item/proto :new mesg))
(yes (send modal-button/proto
:action #'(lambda ()
(setf f (open "RGLM.IN" :direction :output))
(format f "title \"trial\"-\%filexy xy.dat-\%rzes-\%dnost-\%")
(close f)
(system "/home/faculty/rglm/rglm/bin/rglm-f")
(setf run 1))))
(no (send modal-button/proto :new "NO"
:action #'(lambda () (setf run 0))))
(rd-dialog (send modal-dialog/proto
:new (list prompt (list yes no))
:title "RDFFITS etc."
:location (list 76 570))))
(send rd-dialog :modal-dialog)
(run))

(defun remove-sublist (whole no-need)
"Return the sublist of WHOLE with NO-NEED (index of no-need elements) removed."
(let* ((index-whole (iseq (length whole)))
(want (dolist (i no-need index-whole)
(setf index-whole (select index-whole
(which (=/= index-whole i)))))
(select whole want)))

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(defun number-to-string (number)
  "This function transform a list of numbers to a
  list of strings (ie. i=>"1") . This function is
  the counterpart of function string-to-number.
  Please note that \ is a escape character in LISP,
  it cause the next character to be treated as a
  letter rather than for its syntactic purpose."
  (let* ((n (length number))
         (result (repeat "\" n)))
    (do ((i 0 (+ 1 i))) ((>= i n) result)
       (setf (select result i) (num-to-string
                        (select number i))))))

(defun string-to-number (string)
  "This function transform a list of num-string
  (ie. number presented in string form) into a
  list of number. It is the counterpart of
  number-to-string fun"
  (let* ((n (length string))
         (result (repeat 0 n)))
    (do ((i 0 (+ 1 i))) ((>= i n) result)
       (setf (select result i)
               (read (make-string-input-stream
                       (select string i)) nil)))))

(defun mei-message-dialog
  (mesg &key (location (list 10 10)))
  (let* ((text (if (consp mesg) mesg (list mesg)))
         (want (apply #'format nil text))
         (ok (send modal-button-proto :new "OK")))
         (mesg-dialog (send modal-dialog-proto
                        :new (list want ok)
                        :location location)))
  (send mesg-dialog :modal-dialog)))

(defun mei-boxplot
  (data &key position (location (list 55 110))
        (size (list 400 400)) (title "Box Plot")
        (variable-labels (list "X" "Y")))
  "Args: (data &key position (title "\"Box Plot\")
        (location (list 55 110)) (size (list 400 400))"
DATA is a sequence, a list of sequences or a matrix. Makes a boxplot of the sequence or a parallel box plot of the sequences in the list or the COLUMNS of the matrix. This is a modified version of boxplot."

(let ((p (send scatterplot-proto :new 2
        :title title :location location
        :size size
        :variable-labels variable-labels
        :show nil)))
  (setq data
    (cond ((matrixp data) (column-list data))
              ((or (not (listp data)) (numberp (car data)))
               (list data)) (t data)))
  (let ((range (get-nice-range (min data) (max data) 4)))
    (send p :range 1 (nth 0 range) (nth 1 range))
    (send p :y-axis t t (nth 2 range)))

;;; set-up range of x-axis
(if position
  (send p
    :range 0 (1- (min position)) (1+ (max position)))
  (send p :range 0 0 (1+ (length data))))

;;; set-up x-axis
(if position (send p :x-axis t t (+ (length position) 2))
  (send p :x-axis t t 1))
(dotimes (i (length data))
  (if position
    (send p :mei-add-boxplot (nth i data)
      :position (select position i))
    (send p :mei-add-boxplot (nth i data)
      :position (1+ i))))
  (send p :show-window)
p))

(defun 2d-scroll (&key
  (prompt1 "2ml") (prompt2 "2m2")
  (data1 range-m1) (data2 range-m2)
  (want1 m1-d) (want2 m2-d)
Create a dialog for two dimensional scroll bars, eg: (m1,m2).

(let*
  ((prompt-m1 (send text-item,proto :new prompt1))
   (prompt-m2 (send text-item,proto :new prompt2))
   (divide (send text-item,proto :new "------------------------"))
   (seperate (send text-item,proto :new "------------------------"))
   (value-m1 (send edit-text-item,proto :new "" :text-length 5))
   (value-m2 (send edit-text-itemproto :new "" :text-length 5))
   (scroll-m1
    (send mei-sequence-scroll-item,proto :new data1
     :text-item value-m1
     :action #'(lambda (x)
                 (def m1-d x)
                 (format nil "",if" x)
                 (send graph :new-fit data pp m1-d m2-d))))
   (scroll-m2
    (send mei-sequence-scroll-itemproto :new data2
     :text-item value-m2
     :action #'(lambda (x)
                 (def m2-d x)
                 (format nil "",if" x)
                 (send graph :new-fit data pp m1-d m2-d))))
   (cancel (send modal-button-proto :new "Cancel" :action nil))
   (ok (send modal-button-proto :new "OK" :action #'(lambda ()
               (list (send value-m1 :text)
                     (send value-m2 :text)))))
   (mlm2-dialog (send modal-dialogproto :new (list (list prompt-m1 value-m1) scroll-m1 divide
                                                   (list prompt-m2 value-m2) scroll-m2 seperate)
(list ok cancel))
:/title (concatenate 'string
 "To choose ("prompt1 "," prompt2 ")")
:location location
:size size))
(send scroll-m1
 :value (car (which (= data1 want1))))
(send scroll-m2
 :value (car (which (= data2 want2))))
(send m1m2-dialog :modal-dialog))

(defun change-format (real formatted)
 "This function will change a list of REAL number
 to be formatted as FORMATTED."
(let* ((n (length real))
 (tmp (repeat " " n))
 (do ((i 0 (+ 1 i))) ((>= i n) tmp)
 (setf (select tmp i) (format nil formatted (select real i))))
 (string-to-number tmp)))

(defun join-strings (string1 string2)
 "Join two lists of same length of strings to have format as (a,b)"
(let* ((n (length string1))
 (result (repeat " " n))
 (do ((i 0 (+ 1 i))) ((>= i n) result)
 (setf (select result i)
 (concatenate 'string "," (select string1 i)
 ," (select string2 i) ")")))

(defun 1d-scroll
 (&key action (location (list 200 730))
 (display display-run) (choice choice-run)
 (prompt "Run ="))
 "Create a dialog for one-dimensional scroll bar.
 eg: run-selection."
(let* ((prompt-run
 (send text-item-proto :new prompt))
 (value-run
 (send edit-text-item-proto :new ""
 :text-length 3))

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(scroll-run
  (send sequence-scroll-item,proto
   :new display
   :text-item value-run
   :action action))
(cancel (send modal-button,proto
   :new "Cancel" :action nil))
(ok (send modal-button,proto
   :new "OK"
   :action #'(lambda ()
      (list (send value-run :text))))))
(run-dialog (send modal-dialog,proto
   :new (list
      (list prompt-run value-run)
      scroll-run
      (list ok cancel))
   :title "Choice of Run"
   :location location))

;;; setup the initial display value to be choice-run.
;;; where CAR=the 0-th element (ie. the first element of a list)
  (send scroll-run :value (car (which (= display choice))))
  (send run-dialog :modal-dialog))

;;;;;;;;;;;;;; LOG-F Distributions ;;;;;;;;;;;;;;;
(defun logf-quant (p ndf ddf)
  "This function gives the quantile of a log F distribution for the given probability p and degrees of freedom (ndf ddf)."
  (log (f-quant p ndf ddf)))
(defun logf-dens (x ndf ddf)
  "This function gives the density of a log F distribution for the given x and degrees of freedom (ndf ddf)."
  (* (f-dens (exp x) ndf ddf) (exp x)))
(defun logf-cdf (x ndf ddf)
  "This function returns the value of Log-F distribution fun at x."
  (f-cdf (exp x) ndf ddf))
(defun logf-rand (n ndf ddf)
  "This fun returns n random variables generated by log-F dist with (ndf ddf)"
  (log (f-rand n ndf ddf)))

(defun logf-score-dscore (n &key (ndf 2) (ddf 2))
  "This function needs ppoints, logf-quant, logf-dens"
  (let*
    ((p (ppoints n))
     (w (logf-quant p ndf ddf))
     (ew (f-quant p ndf ddf))
     (score (/
                (* ndf ddf (- ew 1))
                (* 2 (+ ddf (* ndf ew))))))
    (dscore (/
              (* (+ ndf ddf) ndf ddf ew)
              (* 4 (logf-dens w ndf ddf)
                   (+ ddf (* ndf ew) 2))))
    (std-dev (standard-deviation score))
    (std-score (/ score std-dev))
    (std-dscore (/ dscore std-dev))
    (list std-score std-dscore)))

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
;; Define some useful methods for objects
(defmeth scatterplot-proto
  :add-2-bp (quant data &key (draw t))
"Draw the QQBP. Here -x and -y refer to boxplot-x
and boxplot-y."

(let*
  ((fiv-x (fivnum quant))
   (q1-x (select fiv-x 1))
   (med-x (select fiv-x 2))
   (q3-x (select fiv-x 3))
   (dq-x (* 1.5 (- q3-x q1-x)))
   (low-x (- q1-x dq-x)) ;; theoretical bound
   (low-x ;; real bound
    (min (select quant (which (< low-x quant))))))
   (high-x (+ q3-x dq-x)) ;; theoretical bound
   (high-x ;; real bound
    (max (select quant (which (> high-x quant))))))
   (below-x (which (> low-x quant)))
   (above-x (which (< high-x quant)))
   (lssuspect-x
    (if below-x (select quant below-x)))
   (hsuspect-x
    (if above-x (select quant above-x)))
   (fiv-y (fivnum data))
   (q1-y (select fiv-y 1))
   (med-y (select fiv-y 2))
   (q3-y (select fiv-y 3))
   (dq-y (* 1.5 (- q3-y q1-y)))
   (low-y (- q1-y dq-y))
   (low-y (min (select data (which (< low-y data))))))
   (high-y (+ q3-y dq-y))
   (high-y (max (select data (which (> high-y data))))))
   (below-y (which (> low-y data)))
   (above-y (which (< high-y data)))
   (lssuspect-y (if below-y (select data below-y)))
   (above-x (which (< high-x data)))
   (lssuspect-x (if below-x (select quant below-x)))
   (hsuspect-y (if above-y (select quant above-y)))
   )

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(hsuspect-y (if above-y (select data above-y)))

(range-x (send self :range 0))
(distance-x
  (- (select range-x 1) (select range-x 0)))
(range-y (send self :range 1))
(distance-y
  (- (select range-y 1) (select range-y 0)))
(wid-x (* .04 distance-y))
(wid-y (* .04 distance-x))
(pos-x
  (+ (select range-y 1) (* .1 distance-y)))
(pos-y
  (+ (select range-x 1) (* .1 distance-x)))
(margin-x (round (* .3 distance-y)))
(margin-y (round (* .3 distance-x)))

;;; Setting up the proper margins to have boxplots properly drawn
(send self :margin 0 margin-y margin-x 0)

;;; For boxplot along X-axis
(send self :add-lines (list low-x low-x)
  (list (- pos-x wid-x) (+ pos-x wid-x)))
(send self :add-lines (list high-x high-x)
  (list (- pos-x wid-x) (+ pos-x wid-x)))
(send self :add-lines
  (list low-x q1-x) (list pos-x pos-x))
(send self :add-lines
  (list q3-x high-x) (list pos-x pos-x))
(send self :add-lines (list q1-x q1-x)
  (list (- pos-x wid-x) (+ pos-x wid-x)))
(send self :add-lines (list med-x med-x)
  (list (- pos-x wid-x) (+ pos-x wid-x)))
(send self :add-lines (list q3-x q3-x)
  (list (- pos-x wid-x) (+ pos-x wid-x)))
(send self :add-lines (list q1-x q3-x)
  (list (- pos-x wid-x) (~- pos-x wid-x)))
(send self :add-lines (list q1-x q3-x)
  (list (+ pos-x wid-x) (+ pos-x wid-x)))
(if hsuspect-x (send self :add-points
  hsuspect-x (repeat pos-x (length hsuspect-x))))
(if lsuspect-x (send self :add-points
lsuspect-x (repeat pos-x (length lsuspect-x))))

;;; For boxplot along Y-axis
(send self :add-lines
(list (- pos-y wid-y) (+ pos-y wid-y))
(list low-y low-y))
(send self :add-lines
(list (- pos-y wid-y) (+ pos-y wid-y))
(list high-y high-y))
(send self :add-lines
(list pos-y pos-y) (list low-y q1-y))
(send self :add-lines
(list pos-y pos-y) (list q3-y high-y))
(send self :add-lines
(list (- pos-y wid-y) (+ pos-y wid-y))
(list q1-y q1-y))
(send self :add-lines
(list (- pos-y wid-y) (+ pos-y wid-y))
(list med-y med-y))
(send self :add-lines
(list (- pos-y wid-y) (+ pos-y wid-y))
(list q3-y q3-y))
(send self :add-lines
(list (- pos-y wid-y) (- pos-y wid-y))
(list q1-y q3-y))
(send self :add-lines
(list (+ pos-y wid-y) (+ pos-y wid-y))
(list q1-y q3-y))
(if hsuspect-y
(send self :add-points (repeat pos-y
(length hsuspect-y)) hsuspect-y))
(if lsuspect-y (send self :add-points
(repeat pos-y (length lsuspect-y)) lsuspect-y))))

(defmeth scatterplot-proto :new-fit
(data pp m1 m2)
"Perform the actions after the quantiles are
updated in score-selection."

;;; update quantiles with current (m1,m2)
(def quantile (rearrange-list
([x] (sort x)))
(data pp m1 m2)
"Perform the actions after the quantiles are
updated in score-selection."
(logf-quant pp m1 m2)
(order (order data)))

;;; update the dynamic QQBP
(send self :clear nil)
(send self :add-points quantile data)
(send self :range 0
  (min quantile) (max quantile))
(send self :range 1 (min data) (max data))
(send self :variable-label 0
  (format nil "Log-F (\(^{-1}\),1f,\(^{-1}\),1f)\(\) m1 m2))
(send self :redraw)
; (send self :abline 2 0)
; (send self :abline -2 0)
(send self :add-2-bp quantile data))

(defmeth scatterplot-proto :ref-line
  (x y &key (draw t))
  "Add a reference line of y=x within data points only."
  (let ((min-xy (max (min x) (min y)))
        (max-xy (min (max x) (max y))))
    (send self :add-lines
      (list min-xy max-xy) (list min-xy max-xy))))

(defmeth scatterplot-proto :mei-add-boxplot
  (y &key (width .4) (position 1) (draw t))
  (unless (= 2 (send self :num-variables))
    (error "only works for 2D plots")
    (let*
      ((fiv (fivnum y))
       (q1 (select fiv 1))
       (med (select fiv 2))
       (q3 (select fiv 3))
       (dq (* 1.5 (- q3 q1)))
       (low (- q1 dq))
       (low (min (select y (which (< low y))))
       (high (+ q3 dq))
       (high (max (select y (which (> high y))))
       (below (which (> low y)))
       (lsuspect (if below (select y below)))
       (above (which (< high y)))
      )
    )
  )
)
(hsuspect (if above (select y above))))
(send self :plotline
  (- position width) low
  (+ position width) low nil)
(send self :plotline
  (- position width) high
  (+ position width) high nil)
(send self :plotline
  position low position q1 nil)
(send self :plotline
  position q3 position high nil)
(send self :plotline
  (- position width) q1
  (+ position width) q1 nil)
(send self :plotline
  (- position width) med
  (+ position width) med nil)
(send self :plotline
  (- position width) q3
  (+ position width) q3 nil)
(send self :plotline
  (- position width) q1
  (- position width) q3 nil)
(send self :plotline
  (+ position width) q1
  (+ position width) q3 nil)
(if hsuspect (send self :add-points
  (repeat position (length hsuspect)) hsuspect))
(if lsuspect
  (send self :add-points
  (repeat position (length lsuspect))
  lsuspect)))

(defun scatterplot_proto
  :add-2-one (quant data &key (draw t))
"Draw the QQBP. Here -x and -y refer to boxplot-x and boxplot-y."

(setf len (length data))
(setf range-x (send self :range 0))
(setf distance-x
  (- (select range-x 1) (select range-x 0)))

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(setq range-y (send self :range 1))
(setq distance-y
     (- (select range-y 1) (select range-y 0)))
(setq pos-x
     (+ (select range-y 1) (* .1 distance-y)))
(setq pos-y
     (+ (select range-x 1) (* .1 distance-x)))
(setq margin-x (round (* .3 distance-y)))
(setq margin-y (round (* .3 distance-x)))

;;; Setting up the proper margins to have
;;; boxplots properly drawn ;;;
(send self :margin 0 margin-y margin-x 0)

;;; For boxplot along X-axis
(send self :add-points quant (repeat pos-x len))
(send self :add-points (repeat pos-y len) data)
;;; rglm-call ;;;;
(require "mei-fun")
(provide "rglm-call")

(defun rglm-call ()
"prepare and run for RGLM."
(setf result (score-setup)) ;log-F scores
(when result
(run-rglm) ;; run RGLM with regular diagnostics
(setf tmp (read-data-columns "RRESID.OUT" 8))

(cond
((= initial-rglm 1)
 (def n-g-all (list n-g))
 (def m1-g-all (list m1-g))
 (def m2-g-all (list m2-g))
 (def case-no-all
 (list (iseq (length (select tmp 0))))))
 (def fitted-all
 (list (copy-list (select tmp 2))))
 (def resid-all
 (list (copy-list (select tmp 3))))
 (def scores-all
 (list (copy-list (select tmp 4))))
 (def leverage-all
 (list (copy-list (select tmp 5))))
 (def stdr-all (list
 (copy-list (select tmp 6))))
 (def quant-all (list
 (rearrange-list
 (logf-quant (ppoints n-g) m1-g m2-g)
 (order (order (select tmp 6)))))
 (def initial-rglm 0))

((= initial-rglm 0)
 (def n-g-all (append n-g-all (list n-g)))
 (def m1-g-all (append m1-g-all (list m1-g)))
 (def m2-g-all
 (append m2-g-all (list m2-g)))
 (def case-no-all (append case-no-all
 (list (iseq (length (select tmp 0))))))
 (def fitted-all (append fitted-all
(list (select tmp 2)))
(def resid-all (append resid-all
(list (select tmp 3)))
(def scores-all (append scores-all
(list (select tmp 4)))
(def leverage-all (append leverage-all
(list (select tmp 5)))
(def stdr-all (append stdr-all
(list (select tmp 6)))
(def quant-all (append quant-all (list
(rearrange-list
(logf-quant (ppoints n-g) m1-g m2-g)
(order (order (select tmp 6))))))))
(def m1-d m1-g)
(def m2-d m2-g)
(def m1-s (list m1-g))
(def m2-s (list m2-g))
(def y-lab (format nil
"R-std residuals (\$-,if,-,if\$"
m1-g m2-g)))

(defun score-setup ()
"This function will setup (n,2m1,2m2), generate
scores, and save them to file scores.dat. It
also update the default value want-n,m1,m2 and
n-g, m1-g, m2-g, n-g-all, m1-g-all, and m2-g-all"
(let*
((label-n (send text-item-PROTO :new "n="))
(label-m1 (send text-item-PROTO :new "2m1="))
(label-m2 (send text-item-PROTO :new "2m2="))
(value-n (send edit-text-item-PROTO
:new (num-to-string want-n)
:text-length 6))
(value-m1 (send edit-text-item-PROTO
:new (num-to-string want-m1)
:text-length 6))
(value-m2 (send edit-text-item-PROTO
:new (num-to-string want-m2)
:text-length 6))
(cancel (send modal-button-PROTO
:new "Cancel" :action nil))
(ok (send modal-button-PROTO
:new "OK" :action #'(lambda ()
  (list (send value-n :text)
    (send value-m1 :text)
    (send value-m2 :text))))
(nmlm2-dialog (send modal-dialog,proto
  :new (list (list label-n value-n)
    (list label-m1 value-m1)
    (list label-m2 value-m2)
    (list ok cancel))
  :title
  "For scores to run RGLM"
  :location (list 400 235)
  :size (list 350 180))
(result
  (send nmlm2-dialog :modal-dialog)))
;;; when nmlm2-dialog return non-nil, then we are
;;; ready to run RGLM
(when result
  (let* ((result (string-to-number result))
    (tmp1 (select result 0))
    (tmp2 (select result 1))
    (tmp3 (select result 2)))
    (setf scores
      (make-array (list tmp1 2)
        :initial-contents (transpose
          (logf-score-dscore tmp1
            :ndef tmp2 :ddf tmp3))))
  (with-open-file
    (f "scores.dat" :direction :output)
    (format f "*****-%*/f~*/.")
    (dotimes (i tmp1)
      (format f "~f~*/f~*/.")
        (aref scores i 0) (aref scores i 1))))
  (def n-g tmp1)
  (def want-n n-g)
  (def m1-g tmp2)
  (def want-m1 m1-g)
  (def m2-g tmp3)
  (def want-m2 m2-g))
(result))
(require "mei-fun")
(require "mei,proto")
(require "mei-meth")
(provide "score-call")

(defun score-call
  (require (quantile-fun #'logf-quant))
"Perform the score-selection for a given set of studentized residuals."

(def num-all 1) ;; initial for number of plots
(def static-plots nil)

(def resid (car (last stdr-all)))
(def pp (ppoints n-g))
(def quantile (rearrange-list
  (funcall quantile-fun pp m1-d m2-d)
  (order (order resid))))

;;; To set up the dynamic QQBP ;;;
(def dynamic
  (plot-points quantile resid
    :title "Dynamic Search"
    :variable-labels (list
      (format nil "Log-F (\"1f\",1f)" m1-d m2-d)
      y-lab)
    :location (list 250 50) ;; (Left,Top)
    :size (list 350 350) ;; (Width,Height)
    :show nil))

(send dynamic :range 0
  (min quantile) (max quantile))
(send dynamic :range 1
  (min resid) (max resid))
(send dynamic :x-axis t t 4)
(send dynamic :y-axis t t 4)
(send dynamic :redraw)
(send dynamic :add-2-bp quantile resid)

;;; To set up the static QQBP ;;;
(def static (plot-points quantile resid
  :title "Static Search"
:variable-labels (list
  (format nil "Log-F (^-1f,-1f)"
   (car (last m1-s)) (car (last m2-s)))
y-lab)
:location (list 650 50)
:size (list 350 350)
:show nil))
(send static :range 0
 (min quantile) (max quantile))
(send static :range 1
 (min resid) (max resid))
(send static :x-axis t t 4)
(send static :y-axis t t 4)
(send static :redraw)
(send static :add-2-bp quantile resid)

(def range-m1
 (sequence-selection
 :string "2M1"
 :low (max .5 (- m1-g 2))
 :middle m1-g
 :high (+ m1-g 2))
(if range-m1
 (def m1-d (select range-m1 19)))
(def range-m2
 (sequence-selection
 :string "2M2"
 :low (max .5 (- m2-g 2))
 :middle m2-g
 :high (+ m2-g 2))
(if range-m2 (def m2-d (select range-m2 19)))
(send dynamic :new-fit resid pp m1-d m2-d)

;;; Dynamic searching for the proper (m1,m2) ;;;
(send dynamic :show-window)
(2d-scroll)
(dynamic-reactions)

;;;;;;; Dynamic QQBP Reactions ;;;;;;;;
(defun dynamic-reactions ()
 "Use a dialog to react actions about dynamic QQBP search: add, change, display and done."

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(let*
  ((dynamic-menu (send choose-item-dialog-proto
    :new "Actions to screen values of (2ml,2m2)"
    (list
      "Add current result to the static search list"
      "Change the search range of (2ml,2m2)"
      "Go to the static search"
      :location '(240 470)
      :title "Dynamic Search of (2ml,2m2)"
      :initial 0))
   (do ((choice 0))
     ((equal choice 2) (dynamic-goes))
     (unless (send dynamic-menu :allocated-p)
       (send dynamic-menu :allocate))
     (setf choice
       (send dynamic-menu :modal-dialog))
     (when choice
       (cond
         ((= choice 0) (dynamic-add))
         ((= choice 1) (dynamic-change)))))

(defun dynamic-add ()
  "Add the current (ml-d, m2-d) to the list for
  Static Search, and display the new image to
  static QQBP"
  (send static :new-fit resid pp ml-d m2-d)
  (def ml-s (change-format
    (append ml-s (list m1-d)) "",lf"))
  (def m2-s (change-format
    (append m2-s (list m2-d)) "",lf"))
  (2d-scroll))

(defun dynamic-change ()
  "Change the range of (ml, m2) for next dynamic
  search."
  (def range-m1 (sequence-selection
    :string "2M1"
    :low (max .5 (- m1-d 2))
    :middle m1-d
    :high (+ m1-d 2))
  (if range-m1 (def m1-d (select range-m1 19))))
(def range-m2 (sequence-selection
 :string "2M2"
 :low (max .5 (m2-d 2))
 :middle m2-d
 :high (+ m2-d 2))
(if range-m2 (def m2-d (select range-m2 19)))
(send dynamic :new-fit resid pp m1-d m2-d)
(2d-scroll))

(defun dynamic-goes ()
 "This function will do the static search"

(send dynamic :hide-window)
(send static :show-window)
(static-reactions))

;;;;;;; ,* ; Static QQBP Reactions ;;;;;;;;
(defun static-reactions ()
 "Use a dialog to react actions about static QQBP
 search: screening, satisfied."

(def want (1- (length m1-s)))
(let ((static-menu (send choose-item -dialog-proto
 :new "Actions to pin down values of (2m1,2m2):
 (list "View results on the static list"
 "Go back to the dynamic search"
 "Done with the search (result=current display)"
 :location ' (660 470)
 :title "Static Search of (2m1,2m2)"
 :initial 0)))

(setf choice-static (loop
 (unless (send static-menu :allocated-p)
 (send static-menu :allocate))
 (setf choice (send static-menu :modal-dialog))
 (cond
 ((equal choice 0) (static-screening))
 ((equal choice 1) (return choice))
 ((equal choice 2) (return choice))))))

(cond
((= choice-static 1) (static-goes))
((= choice-static 2) (static-satisfied)))

(defun static-screening ()
"Set up the style of display for screening."

(let*
((screening-style-menu
  (send choose-item-dialog-proto
    :new
    "What Style of Display Do You Like to See?"
    (list "display all at once"
      "display one at a time"
      "quit")
    :title "Choice of Styles"
    :initial 0
    :location (list 660 470)
    :show nil)))

(do
  ((choice 0)) ;; initialize/update
  ((equal choice 2)) ;; exit condition
  (unless
    (send screening-style-menu :allocated-p)
    (send screening-style-menu :allocate))
  (setf choice
    (send screening-style-menu :modal-dialog))
  (when (and choice (< choice 2))
    (cond
      ((= choice 0) (static-display-all))
      ((= choice 1) (static-display-one)))))

(defun static-goes ()
"This function will do the dynamic search"

(send static :hide-window)
(send dynamic :show-window)
(2d-scroll)
(dynamic-reactions))

(defun static-display-all ()
"Display all the chosen (m1,m2) QQBP."
(send static :hide-window)
(let*
  ((n (length m1-s))
   (plot-loc (list (list 20 20) (list 350 20)
                  (list 680 20) (list 20 400)
                  (list 350 400) (list 680 400)))
   (def num-all n)
   (def static-plots (repeat nil n))
   (if (> n 6)
       (mei-message-dialog
        "We can only plot up to 6 plots at a time"
        :location '(400 400)))
   (dotimes (i num-all)
     (setf tmp-m1 (select m1-s i))
     (setf tmp-m2 (select m2-s i))
     (setf tmp (rearrange-list
                (logf-quant pp tmp-m1 tmp-m2)
                (order (order resid))))
     (setf (select static-plots i)
           (plot-points tmp resid
                        :title (concatenate 'string "Plot 
                                    (num-to-string i))
                        :variable-labels (list
                                          (format nil "Log-F (",lf," ,lf)"
                                                  tmp-m1 tmp-m2)
                                          y-lab)
                        :location (select plot-loc (mod i 6))
                        :size (list 300 300)))
     (send plot :range 0 (min tmp) (max tmp))
     (send plot :range 1
                (min resid) (max resid))
     (send plot :x-axis t t 4)
     (send plot :y-axis t t 4)
     (send plot :redraw)
     (send plot :add-2-bp tmp resid)
     (if (= (mod i 6) 5)
       (mei-message-dialog
        "See the plots")
(defun static-display-one ()
"Display QQBP of chosen (m1,m2) one at a time."
(let*
  ((n (length m1-s))
   (m1-ss (number-to-string m1-s))
   (m2-ss (number-to-string m2-s))
   (m12-s (join-strings m1-ss m2-ss))
   (tmp (list "Quit"))
   (m12-s (append m12-s tmp))
   (screening-dialog
    (send choose-item-dialog-proto
     :new "List of Chosen (m1,m2)" m12-s
     :location '(680 470)
     :title "To see the chosen result"
     :initial (- n 1)
     :show nil)))
(def want (1- n))
(do
  ((choice (- n 1))) ;; exit condition
    (equal choice n))
(unless
  (send screening-dialog :allocated-p)
  (send screening-dialog :allocate))
(setf choice
  (send screening-dialog :modal-dialog))
(when (and choice (< choice n))
  (send static :new-fit resid pp
    (select m1-s choice) (select m2-s choice))
  (def want choice))))

(defun static-satisfied ()
"close window of static QQBP and save the final
(m1,m2) to (want-m1,want-m2)."
(send static :close)
(send dynamic :close)
(if (> num-all i) (do times (i num-all)
  (send (select static-plots i) :close)))
(def want-m1 (select m1-s want))
(def want-m2 (select m2-s want))
(require "mei-fun")
(require "mei-meth")
(require "rglm-call")
(provide "diag-call")

(defun diag-call ()
"Reactions to the diagnostics call."

(setup-data)
(plot-resids)

(let
  ((diag-menu
    (send choose-item-dialog-proto
      :new "Actions:"
      (list "Display various RGLM results" "Done")
      :title "Choice of Actions"
      :initial 0)))

  (do
    ((diag-choice 0))
    ((equal diag-choice 1) ())
    (unless (send diag-menu :allocated-p)
      (send diag-menu :allocate))
    (setf diag-choice
      (send diag-menu :modal-dialog))
    (when diag-choice
      (if (= diag-choice 0) (diag-display)))))

(defun diag-display ()
"Display residual plots."

(let
  ;; dialog for residual plots
  ((display-menu (send choose-item-dialog-proto
    :new ""
    (list "All residual plots"
      "Individual residual plot"
      "Boxplot of studentized residuals"
      "None")))

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(do
  ((display-choice 0)) ;; initialize/update
  ((equal display-choice 3) ()) ;; exit condition
  (unless (send display-menu :allocated-p)
    (send display-menu :allocate))
  (def display-choice
    (send display-menu :modal-dialog))
  (when display-choice
    (cond
      ;; all except RDFFITS & R-EXT
      ((= display-choice 0) (display-all))
      ;; individual ones
      ((= display-choice 1) (display-ind))
      ;; boxplot of std. resid for all
      ((= display-choice 2) (display-bp))))))

(defun display-all ()
  "Plot all the 6 residual plots except of RDFFITS and R-EXT."

  (dotimes (i 6) (send (select resid-plots i)
    :show-window))
  (let
    ((outlier-menu (send choose-item-dialog-proto
      :new
      "Types of Outliers:
      (list "Studentized residuals"
        "Leverage values"
        "Combine all of the above"
        "Quit")
      :title "Choices of Outliers"
      :location (list 400 700)
      :initial 0)))
    (do
      ((out-choice 0))
      ((equal out-choice 3) ())
      (def out-choice (send outlier-menu
        :modal-dialog))
      (when (and out-choice (< out-choice 3))
(cond
  ((= out-choice 0)
   (def outliers-diag outliers-r))
  ((= out-choice 1)
   (def outliers-diag outliers-l))
  ((= out-choice 2)
   (def outliers-diag outliers-all))
  (send (select resid-plots 2) :selection
     (select outliers-diag choice-run)))
(cond
  ((= num-run 1)
   (mei-message-dialog "See the plots?"
     :location (list 200 730)))
  (> num-run 1)
   (def choice-run (select (string-to-number
     (id-scroll :action #'(lambda (x)
         (update-plots x)))) 0))
   (def index-case-no
     (select case-no-all choice-run))
   (def display-case-no
     (number-to-string index-case-no)))
  (dotimes (i 6) (send (select resid-plots i)
     :hide-window)))

(defun display-ind ()
"Display residual plots individually."

(let
  ;; which residual plot?
  ((ind-menu (send choose-item-dialog-proto
     :new ""
     (list "Fitted value vs. studentized residuals"
         "Case number vs. studentized residuals"
         "QQBP of studentized residuals"
         "Fitted value vs. residuals"
         "Case number vs. leverage values"
         "Case number vs. scores"
         "None")
     :title "Choice of Residual Plots"
     :initial 0
     :show nil))
  (ind-style-menu (send choose-item-dialog-proto

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(do
  ((ind-choice 0)) ;; initialize/update
  ((equal ind-choice 6) ()) ;; exit condition
  (unless (send ind-menu :allocated-p)
    (send ind-menu :allocate))
  (setf ind-choice (send ind-menu :modal-dialog))
  (when (and ind-choice (< ind-choice 6))
    (unless (send ind-style-menu :allocated-p)
      (send ind-style-menu :allocate))
    (setf ind-style-choice
      (send ind-style-menu :modal-dialog))
    (cond
      ((= ind-style-choice 0) (ind-all ind-choice))
      ((= ind-style-choice 1)
        (ind-movie ind-choice))))))

(defun ind-all (index-p)
  "We display the chosen residual plots 6 at a time together."

  (let ((tmp (repeat nil num-run)))
    (do ((i num-run)
         (i+ i num-run)
         (if (= index-p 2)
           (setf (select plot-title index-p)
             (format nil "(m1,m2)=" (select m1-g-all i) (select m2-g-all i)))
           (setf (select tmp i)
             (plot-points
               (select (select
                 (select plot-data index-p) 0) i)
               (select (select
                 (select plot-data index-p) 1) i)
               :title
               (concatenate 'string
                 (select plot-title index-p)
                 (num-to-string i) "))))
    ))
:variable-labels (select plot-var-lab index-p)
:location (select plot-loc (mod i 6))
:size (list 280 280))
(setf plot (select tmp i))
(send plot :showing-labels t)
(setf tmp-ind (iseq (select n-g-all i)))
(send plot :point-label
  tmp-ind (number-to-string tmp-ind))
(send plot :linked t)
(send plot :selection (select outliers-diag i))

(cond
  ((or (= index-p 0) (= index-p 1))
    (send plot :abline 2 0)
    (send plot :abline -2 0))
  ((= index-p 4)
    (send plot :abline (* 2 (/ p-g n-g)) 0))
  ((= index-p 2)
    (send plot :abline 2 0)
    (send plot :abline -2 0)
    (send plot :add-2-bp
      (select (select
        (select plot-data index-p) 0) i)
      (select (select
        (select plot-data index-p) 1) i))))
  (if (or (= i (1- num-run)) (= (mod i 6) 5))
    (mei-message-dialog "Individual Plot"
      :location (list 950 150))))
(dotimes (i num-run)
  (send (select tmp i) :close)))

(defun ind-movie (index-p)
  "We display the chosen residual plot in movie-like style."

(let ((plot (select resid-plots index-p)))
  (send plot :location 420 275)
  (send plot :show-window)
  (cond
    ((= num-run 1)
      (mei-message-dialog "See the plot?"
        :location (list 450 600)))

(send plot :hide-window))
((> num-run 1)
 (id-scroll
   :location (list 450 600)
   :action #'(lambda (x)
      (setf tmp-ind (iseq (select n-g-all x)))
      (send plot :clear nil)
      (setf tmp-x (select (select plot-data index-p) 0) x))
      (setf tmp-y (select (select plot-data index-p) 1) x))
      (send plot :add-points tmp-x tmp-y)
      (send plot :range 0 (min tmp-x) (max tmp-x))
      (send plot :range 1 (min tmp-y) (max tmp-y))
      (send plot :point-label tmp-ind
        (number-to-string tmp-ind))
      (send plot :showing-labels t)
      (send plot :selection
        (select outliers-diag x))
      (cond
        ((or (= index-p 0) (= index-p 1))
          (send plot :abline 2 0)
          (send plot :abline -2 0))
        ((= index-p 4)
          (send plot :abline (* 2 (/ p-g n-g)) 0))
        ((= index-p 2)
          (send plot :abline 2 0)
          (send plot :abline -2 0)
          (send plot :add-2-bp
            (select (select
              (select plot-data index-p) 0) x)
            (select (select
              (select plot-data index-p) 1) x))))
      (send plot :title (concatenate 'string
        (select plot-title index-p)
        (num-to-string x) "")
      (if (= index-p 2)
        (setf (select plot-title index-p)
          "(m1,m2)=","1f","1f") "
          (select m1-g-all x)
          (select m2-g-all x)))
      (send plot :redraw)))))))
(send plot :hide-window)
(send plot :location
  (select (select plot-loc index-p) 0)
  (select (select plot-loc index-p) 1)))

(defun display-bp ()
  "Plot the boxplot of studentized residuals for all RGLM runs."
  (def all-r
    (mei-boxplot (select (select plot-data 0) 1)
      :position display-run
      :variable-labels (list "RGLM runs"
                          "studentized residuals")
      :location (list 100 100)
      :size (list 400 400)))
  (send all-r :abline -2 0)
  (send all-r :abline 2 0)
  (mei-message-dialog
   "Boxplot of std. resid. for all RGLM runs"
   :location (list 95 590))
  (send all-r :hide-window))

(defun setup-data ()
  "Set up data for R-diagnostics."

  (run-setup)
  ;; set up (data, title, location, and var-lab)
  ;; for the 6 residual plots
  (def plot-data (list
                   (list fitted-all stdr-all)
                   (list case-no-all stdr-all)
                   (list quant-all stdr-all)
                   (list fitted-all resid-all)
                   (list case-no-all leverage-all)
                   (list case-no-all scores-all)))

  (def plot-title (list
                   "Fitted vs. Std Resid ("
                   "Case # vs. Std Resid ("
                   (format nil "'(m1,m2)=\((\",lf,\",\",lf)\) ("
                       (select m1-g-all choice-run)
                       (select m2-g-all choice-run))
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
"Fitted vs. Residuals ("
"Case No. vs. Leverage ("
"Case No. vs. Scores ("))
(def plot-var-lab (list
(list "fitted values" "studentized residuals")
(list "case number" "studentized residuals")
(list "Log-F" "studentized residuals")
(list "fitted values" "residuals")
(list "case number" "leverage")
(list "case number" "scores")))
(def plot-loc (list (list 20 35) (list 330 35)
(list 650 35) (list 20 400) (list 330 400)
(list 650 400)))

(defun run-setup ()
"Setup proper information about run."
(def num-run (length n-g-all))
;;; default run=last
(def choice-run (1- num-run))
(def index-case-no
(select case-no-all choice-run))
(def display-case-no
(number-to-string index-case-no))
;;; using XLS index no. (started 0)
(def display-run (iseq num-run))

;;; set up the OUTLIERS
(def outliers-r (repeat nil num-run))
(def outliers-1 (repeat nil num-run))
(def outliers-all (repeat nil num-run))
(dotimes (i num-run)
(let
(((r-out1 (which (< (select stdr-all i) -2)))
(r-out2 (which (> (select stdr-all i) 2)))
(l-out (which (> (select leverage-all i)
(* 2 (/ p-g n-g))))) ;; leverage
(setf (select outliers-r i)
(remove-duplicates (append r-out1 r-out2)))
(setf (select outliers-1 i) l-out)
(setf (select outliers-all i)
(remove-duplicates (append
   (select outliers-r i)
   (select outliers-l i))))
;;; default setting
(def outliers-diag outliers-r))

(defun plot-resids ()
"Set up all the residual plots without showing
them."

(let
 ((run-string (num-to-string choice-run)))
 (def resid-plots (repeat nil 6))
 (dotimes (i 6)
   (setf (select resid-plots i)
      plot-points
      (select (select (select plot-data i) 0)
         choice-run)
      (select (select (select plot-data i) 1)
         choice-run)
      :title (concatenate 'string
         (select plot-title i) run-string "")
      :variable-labels (select plot-var-lab i)
      :shown nil))
 (setf plot (select resid-plots i))
 (send plot :location
    (select (select plot-loc i) 0)
    (select (select plot-loc i) 1))
 (send plot :size 280 280)
 (send plot :showing-labels t)
 (send plot :point-label index-case-no
     display-case-no)
 (send plot :linked t)
 (send plot :selection
    (select outliers-diag choice-run))

(cond
 ((or (= i 0) (= i 1))
  (send plot :abline 2 0)
  (send plot :abline -2 0))
 ((= i 4)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(send plot :abline (* 2 (/ p-g n-g)) 0))
((= i 2)
 (send plot :abline 2 0)
 (send plot :abline -2 0)
 (send plot :add-2-bp
     (select (select (select plot-data i) 0)
             choice-run)
     (select (select (select plot-data i) 1)
             choice-run))))

(defun update-plots (x)
  "Update all 6 residual plots with run=x data."

  (dotimes (i 6)
    (setf plot (select resid-plots i))
    (send plot :clear nil)
    (send plot :add-points
      (select (select (select plot-data i) 0) x)
      (select (select (select plot-data i) 1) x))
    (if (= i 2)
      (setf (select plot-title i)
        (format nil "(m1,m2)=(-,1f,-,1f) ("
            (select m1-g-all x) (select m2-g-all x))))
      (send plot :title
        (concatenate 'string (select plot-title i)
         (num-to-string x) ")))
    (send plot :point-label
      (iseq (select n-g-all x))
      (number-to-string (iseq (select n-g-all x))))
    (send plot :showing-labels t)
    (send plot :linked t)
    (send plot :selection (select outliers-diag x))
    (cond
      ((or (= i 0) (= i 1))
       (send plot :abline 2 0)
       (send plot :abline -2 0))
      ((= i 4)
       (send plot :abline (* 2 (/ p-g n-g)) 0))
      ((= i 2)
       (send plot :abline 2 0)
       (send plot :abline -2 0)
       (send plot :add-2-bp
        ...
(select (select (select plot-data i) 0) x)
(select (select (select plot-data i) 1) x)))
(send plot :redraw))
(require "mei-fun")
(require "mei-meth")
(require "rglm-call")
(provide "rd-call")

(defun rd-call ()
  "Reactions to the RDFFITS/R-EXT call."

;;; keep original information
(defun original-data-r (read-data-r "xy.dat")
  (def original-n
    (select (select original-data-r 0) 0))
  (def original-p
    (select (select original-data-r 0) 1))
  (def original-p1 (1+ original-p))
  (def original-case-no (iseq original-n))
  (def original-case-lab
    (number-to-string original-case-no))

  (def rd-want-n original-n)
  (def rd-want-m1 2)
  (def rd-want-m2 2)

;;; initiate the value of first-rd
(def first-rd 1) ; ;complement to initial-rd

;;; dialog for RDFFITS
(let ((rd-menu (send choose-item-dialog-proto
      :new ""
      (list
        "Run RGLM to get RDFFITS and R-EXT statistics"
        "Display results"
        "Delete outliers to refit"
        "Done")
      :title "Choice of Actions"
      :initial 0)))

  (do
    ((rd-choice 0)) ;; initialize/update
    ((equal rd-choice 3)
      (write-data-r original-data-r "xy.dat")))
(unless (send rd-menu :allocated-p)
  (send rd-menu :allocate))
(def rd-choice (send rd-menu :modal-dialog))
(when rd-choice
  (cond
    ;; action=obtain RDFFITS/R-EXT
    ((= rd-choice 0) (rd-run))
    ;; action=display results
    ((= rd-choice 1) (rd-display))
    ;; action=delete outliers to refit
    ((= rd-choice 2) (rd-delete))))))

(defun rd-run ()
"Run RGLM to obtain RDFFITS and R-EXT statistics
then save them. Before we run RGLM, we ask user
to specify scores (Log Paretos in RGLM)."

(setf run 0) ;;; initiate
(setf run (rd-confirm))
(when (= run 1) ;; ie. want to run RGLM
  (rd-score-setup) ;;Log Paretos scores, run RGLM
  (setf tmp1 (read-data-columns "fort.52" 6))
  (setf tmp2 (read-data-columns "RRESID.OUT" 7))
  (cond
    ((= initial-rd 1)
      (def rd-n-all (list rd-want-n))
      (def rd-m1-all (list rd-want-m1))
      (def rd-m2-all (list rd-want-m2))
      (def rd-all
       (list (copy-list (select tmp1 3))))
      (def ext-all
       (list (copy-list (select tmp1 2))))
      (def rd-fitted-all
       (list (copy-list (select tmp2 2))))
      (def rd-case-no-all (list original-case-no))
      (def rd-case-lab-all (list original-case-lab))
      (def rd-resid-all
       (list (copy-list (select tmp2 3))))
      (def rd-scores-all
       (list (copy-list (select tmp2 4))))
      (def rd-leverage-all
       (list (copy-list (select tmp2 5)))))))
(def rd-stdr-all (list
    (copy-list (select tmp2 6)))))
(def rd-quant-all (list
    (rearrange-list
    (logf-quant
        (ppoints rd-want-n) rd-want-m1 rd-want-m2)
        (order (order (select tmp2 6))))))
(setf need-setup-plot 1)
(def initial-rd 0)

((= initial-rd 0)
    (def rd-n-all (append rd-n-all
        (list rd-want-n)))
    (def rd-m1-all
        (append rd-m1-all (list rd-want-m1)))
    (def rd-m2-all
        (append rd-m2-all (list rd-want-m2)))
    (def rd-all (append rd-all (list
        (select tmp1 3))))
    (def next-all (append next-all (list
        (select tmp1 2))))
    (def rd-fitted-all (append rd-fitted-all
        (list (select tmp2 2))))
    (def rd-case-no-all (append rd-case-no-all
        (list (iseq rd-want-n))))
    (def rd-resid-all (append rd-resid-all
        (list (select tmp2 3))))
    (def rd-scores-all (append rd-scores-all
        (list (select tmp2 4))))
    (def rd-leverage-all (append rd-leverage-all
        (list (select tmp2 5))))
    (def rd-stdr-all (append rd-stdr-all
        (list (select tmp2 6))))
    (def rd-quant-all (append rd-quant-all
        (list (rearrange-list
        (logf-quant
        (ppoints rd-want-n) rd-want-m1 rd-want-m2)
        (order (order (select tmp2 6))))))))
(rd-setup-data)
(when (= need-setup-plot 1)
    (rd-plots-setup)
    (setf need-setup-plot 0)))
(defun rd-display ()
"Display the 8 residual plots for various RGLM runs with choice of outliers and runs"

(if (= initial-rd 1)
  (mei-message-dialog
   "Please run RGLM first to get statistics"
   :location (list 200 100)))

(when (= initial-rd 0)
  (rd-update-plots rd-choice-run)
  (dotimes (i 8)
    (send (select rd-plots i) :show-window))
  (let
    ((outlier-menu (send choose-item-dialog-proto
                      :new "Types of Outliers:"
                      (list "Studentized Residuals"
                            "Leverage Values"
                            "RDFFITS"
                            "R-EXT"
                            "Combine all the above"
                            "Quit")
                      :title "Choices of Outliers"
                      :location (list 400 650)
                      :initial 0)))
    (do ((rd-out-choice 0))
        ((equal rd-out-choice 5) ()
         (def rd-out-choice (send outlier-menu :
                               modal-dialog))
        (when (and rd-out-choice (< rd-out-choice 5))
          (cond
           ((= rd-out-choice 0)
            (def rd-out rd-out-r))
           ((= rd-out-choice 1)
            (def rd-out rd-out-l))
           ((= rd-out-choice 2)
            (def rd-out rd-out-rd))
           ((= rd-out-choice 3)
            (def rd-out rd-out-rext))
           ((= rd-out-choice 4)
            (def rd-out rd-out-all)))))}
(send (select rd-plots 4) :selection
  (select rd-out rd-choice-run)))
(cond
  ((= rd-num-run 1)
   (mei-message-dialog "See the plots?"
     :location (list 200 730)))
  (> rd-num-run 1)
  (def rd-choice-run (car (string-to-number
    (id-scroll
     :action '#'(lambda (x) (rd-update-plots x))
     :display rd-display-run
     :choice rd-choice-run)))
  (def rd-index-case-no
    (select rd-case-no-all rd-choice-run))
  (def rd-display-case-no
    (number-to-string rd-index-case-no)))
  (dotimes (i 8)
    (send (select rd-plots i) :hide-window))))

(defun rd-delete ()
"Perform outliers deletion."

(if (= initial-rd 1)
  (mei-message-dialog
    "Please run RGLM first to get statistics"
    :location (list 200 100)))

; show plots with all the outlying points marked
(def rd-out rd-out-all)
(dotimes (i 8)
  (send (select rd-plots i) :selection
    (select rd-out (i- rd-num-run)))
  (send (select rd-plots i) :show-window))

; give instructions to choose the "outliers"
  (mei-message-dialog
    "To select points, eg: point 1, 2 and 3, type in \"(list 1 2 3)\".
    To select NO point, click \"Cancel\" button."
    :location (list 300 660))

(setf delete-temp (get-value-dialog
  "Expression for indices to select:"
(:location (list 300 660)))

(send (select rd-plots 4)
 :selection (car delete-tmp))
(setf del 0) ;; del=index for deletion or not
(let*
  ((mesg
     "Do you really want to delete those points?")
   (yes (send modal-button,proto :new "Yes"
       :action #'(lambda () (setf del 1))))
   (no (send modal-button,proto
        :new "No" :action nil))
   (del-dialog (send modal-dialog,proto
                     :new (list mesg (list yes no))
                     :title "To Confirm Choice of Points"
                     :location (list 220 660)))
   (send del-dialog :modal-dialog)
   (when (= del 1)
     ;; delete points and update file "xy.dat"
     ;; rd-del-orig = in original index (for lab)
     ;; rd-del-curr = in current index (for :select)
   (cond
     ((= first-rd 1)
      (def rd-del-orig (copy-list delete-tmp))
      (def rd-del-curr (copy-list delete-tmp))
      (def rd-case-lab-all (append rd-case-lab-all
                               (list (number-to-string
                                       (remove-sublist
                                        (string-to-number
                                         (car (last rd-case-lab-all))
                                         (car delete-tmp))))))
      (def first-rd 0))
     ((= first-rd 0)
      (def rd-del-orig
       (append rd-del-orig delete-tmp))
      (def rd-del-curr (append rd-del-curr
                               (list (match (car delete-tmp)
                                       (string-to-number
                                        (car (last rd-case-lab-all))))))
      (def rd-case-lab-all (append rd-case-lab-all
                               (list (number-to-string
                                       (remove-sublist
                                       (string-to-number
                                        (car (last rd-case-lab-all))))))))
      (def first-rd 0))
   )))

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(def rd-want-n
  (- rd-want-n (length (car delete-tmp))))
(setf xy
  (select (select original-data-r 1)
    (string-to-number
      (car (last rd-case-lab-all)))
    (iseq original-p1)))
(with-open-file
  (f "xy.dat" :direction :output)
  (format f "*****~%")
  (format f "~d "d "~%" rd-want-n original-p)
  (dotimes (i rd-want-n)
    (dotimes (j original-p1)
      (format f "~f " (aref xy i j)))
    (format f "~%")))
  (dotimes (i 8) (send (select rd-plots i)
    :hide-window))))

(defun rd-update-plots (x)
  "Update all 8 residual plots with run=x data."
  (dotimes (i 8)
    (setf plot (select rd-plots i))
    (send plot :clear nil)
    (send plot :add-points
      (select (select (select rd-plot-data i) 0) x)
      (select (select (select rd-plot-data i) 1) x))
    (if (= i 4)
      (setf (select rd-plot-title i)
        (format nil "(m1,m2)=(~,1f,",1f) (" 
          (select rd-m1-all x)
          (select rd-m2-all x)))))
      (send plot :title (concatenate 'string
        (select rd-plot-title i)
        (num-to-string x) ")")
    (send plot :showing-labels t)
    (send plot :point-label
      (select rd-case-no-all x))
    (send plot :showing-labels t))
  (send plot :showing-labels t))
(select rd-case-lab-all x))
(send plot :linked t)
(send plot :selection (select rd-out x))

(cond
  ((or (= i 0) (= i 2))
   (send plot :abline 2 0)
   (send plot :abline -2 0))
  ((= i 3)
   (send plot :abline (* 2 (/ original-p
                       (select rd-n-all x))) 0))
  ((= i 4)
   (send plot :abline 2 0)
   (send plot :abline -2 0)
   (send plot :add-2-bp
              (select (select
                        (select rd-plot-data i) 0) x)
              (select (select
                        (select rd-plot-data i) 1) x)))
  ((= i 6)
   (send plot :abline
              (* -2 (sqrt (/ original-p
                           (select rd-n-all x)))) 0)
   (send plot :abline
              (* 2 (sqrt (/ original-p
                          (select rd-n-all x)))) 0))
  ((= i 7)
   (send plot :abline 2 0)
   (send plot :abline -2 0))
  (send plot :redraw))))

(defun rd-confirm ()
  "This function will return confirmation about
  the RGLM run with dnost option. 1=yes, 0=no"

(let*
((mesg (format nil "Do you really want to
               obtain RDFITTS statistics?"%(NOTE: This will
               take a while, please be PATIENT)"))
  (prompt (send text-item/proto :new mesg))
  (yes (send modal-button/proto
:new "YES"
:action #'(lambda () (setf run 1)))
(no (send modal-button-proto
 :new "NO"
 :action #'(lambda () (setf run 0))))
(rd-dialog (send modal-dialog-proto
 :new (list prompt (list yes no))
 :title "RDFITS etc."
 :location (list 76 570)))
(send rd-dialog :modal-dialog)
(run))

(defun rd-setup-data ()
 "Setup plot data, title, location, and var-lab information."

(rd-run-setup)
(defun rd-plot-data (list
 (list rd-fitted-all rd-stdr-all)
 (list rd-fitted-all rd-resid-all)
 (list rd-case-no-all rd-stdr-all)
 (list rd-case-no-all rd-leverage-all)
 (list rd-quant-all rd-stdr-all)
 (list rd-case-no-all rd-scores-all)
 (list rd-case-no-all rd-all)
 (list rd-case-no-all rext-all)))
(defun rd-plot-title (list
 "Fitted vs. Std Resid. ("
 "Fitted vs. Resid. ("
 "Case vs. Std Resid. ("
 "Case vs. Leverage ("
 (format nil "(2ml,2m2)=\(",1f",1f) (" 
 (select rd-m1-all rd-choice-run)
 (select rd-m2-all rd-choice-run))
 "Case vs. Scores ("
 "Case vs. RDFITS ("
 "Case vs. R-EXT ("))
(defun rd-plot-varlab (list
 (list "fitted values" "studentized residuals")
 (list "fitted values" "residuals")
 (list "case number" "studentized residuals")
 (list "case number" "leverage")

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(list "Log-F" "studentized residuals")  
(list "case number" "scores")  
(list "case number" "RDFITS")  
(list "case number" "R-EXT"))

(def rd-plot-loc (list (list 10 40)  
(list 10 360) (list 290 40)  
(list 290 360) (list 570 40)  
(list 570 360) (list 850 40) (list 850 360)))

(defun rd-run-setup ()  
"setup proper information about runs."

(def rd-num-run (length rd-n-all))  
(def rd-choice-run (1- rd-num-run))  
(def rd-case-no-index  
(select rd-case-no-all rd-choice-run))  
(if (/= (length rd-case-no-all)  
(length rd-case-lab-all))  
(def rd-case-lab-all (append rd-case-lab-all  
(list (car (last rd-case-lab-all)))))))  
(def rd-case-no-lab  
(select rd-case-lab-all rd-choice-run))  
(def rd-display-run (iseq rd-num-run))

;;; setup outliers

(def rd-out-r (repeat nil rd-num-run))  
(def rd-out-l (repeat nil rd-num-run))  
(def rd-out-1 (repeat nil rd-num-run))  
(def rd-out-rext (repeat nil rd-num-run))  
(def rd-out-all (repeat nil rd-num-run))

(dotimes (i rd-num-run)

(let*  
((r-out1 (which (< (select rd-stdr-all i) -2)))  
(r-out2 (which (> (select rd-stdr-all i) 2)))  
(l-out (which (> (select rd-leverage-all i)  
(* 2 (/ original-p  
(select rd-n-all rd-choice-run)))))  
(rd-out1 (which (< (select rd-all i)  
(* -2 (sqrt (/ original-p  
(select rd-n-all rd-choice-run)))))))  
(rd-out2 (which (> (select rd-all i)  
(* 2 (sqrt (/ original-p  
(select rd-n-all rd-choice-run))))))))
(ext-out1 (which (< (select ext-all i) -2)))
(ext-out2 (which (> (select ext-all i) 2)))

(setf (select rd-out-r i)
   (remove-duplicates (append r-out1 r-out2)))
(setf (select rd-out-l i) l-out)
(setf (select rd-out-rd i)
   (remove-duplicates (append rd-out1 rd-out2)))
(setf (select rd-out-rext i)
   (remove-duplicates
      (append ext-out1 ext-out2)))
(setf (select rd-out-all i)
   (remove-duplicates (append
      (select rd-out-r i)
      (select rd-out-l i)
      (select rd-out-rd i)
      (select rd-out-rext i)))))
(def rd-out rd-out-r) ;; default setting

(defun rd-plots-setup ()
  "setup all the plots without showing them."
  (def rd-plots (repeat nil 8))
  (dotimes (i 8)
    (setf (select rd-plots i)
      (plot-points
       (select (select (select rd-plot-data i) 0)
      rd-choice-run)
       (select (select (select rd-plot-data i) 1)
      rd-choice-run)
       :title (concatenate 'string
      (select rd-plot-title i)
      (car (number-to-string (list rd-choice-run)))
      "")
       :variable-labels (select rd-plot-varlab i)
      :show nil))
    (setf plot (select rd-plots i))
    (send plot :location
      (select (select rd-plot-loc i) 0)
      (select (select rd-plot-loc i) 1))
    (send plot :size 250 250)
    (send plot :showing-labels t)
    (send plot :point-label
(select rd-case-no-all rd-choice-run)
(select rd-case-lab-all rd-choice-run))
(send plot :linked t)
(send plot :selection
(select rd-out rd-choice-run))
(cond
((or (= i 0) (= i 2))
(send plot :abline 2 0)
(send plot :abline -2 0))
((= i 3)
(send plot :abline (* 2 (/ original-p
(select rd-n-all rd-choice-run))) 0))
((= i 4)
(send plot :abline 2 0)
(send plot :abline -2 0)
(send plot :add-2-bp
(select (select (select rd-plot-data i) 0) rd-choice-run)
(select (select (select rd-plot-data i) 1) rd-choice-run)))
((= i 6)
(send plot :abline
(* -2 (sqrt (/ original-p
(select rd-n-all rd-choice-run)))) 0)
(send plot :abline
(* 2 (sqrt (/ original-p
(select rd-n-all rd-choice-run)))) 0))
((= i 7)
(send plot :abline 2 0)
(send plot :abline -2 0)))
(send plot :redraw)))

(defun rd-score-setup ()
"To setup scores for use in RGLM. The scores
used is provided in RGLM and called Log Paretos.
They are acturally either log-F(2,2m2) or
log-F(2m1,2). That is, with one end fixed at 2."

(let*
((score-menu (send choose-item-dialog-proto
:new "Choice of Scores")

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(list "log-F(2,2) scores"
    "log-F(2m1,2) scores"
    "log-F(2,2m2) scores"
    "none")
:location (list 400 400)
:title "Score Menu"
:initial 0))
(setf choice (send score-menu :modal-dialog))
(when choice
  (cond
    ((= choice 0) (rd-parameter :choice 0))
    ((= choice 1) (rd-parameter :choice 1))
    ((= choice 2) (rd-parameter :choice 2))))

(defun rd-parameter (&key (choice 0))
  "This function will setup value that will be used in RGLM to determine the Log Paretos scores."

  (if (= initial-rd 0)
    (def rd-resid (car (last rd-stdr-all)))
    (cond
      ((= choice 0) ;; use default scores
        (def rd-want-m1 2)
        (def rd-want-m2 2)
        (rd-rglm))
      ((= choice 1) ;; use log-F(2m1,2) scores
        (cond
          ((= initial-rd 1) ;; before the first run
            (setf tmp-2m1 (get-1-value
                :label "2m1 ="
                :value rd-want-m1
                :title "log-F(2m1,2) scores"
                :location (list 400 500)))
            (when tmp-2m1
              (setf v-par (/ tmp-2m1 2))
              (def rd-want-m1 tmp-2m1)
              (def rd-want-m2 2)
              (rd-rglm :choice 1 :parameter v-par)))
          ((= initial-rd 0) ;; after the first run
            (rd-setup-search :choice 1)
            (do
              ((change 1))
            )))
    )))
((equal change 0) ())
(def rd-range-m1 (sequence-selection
 :string "2M1"
 :low (max .5 (- rd-want-m1 2))
 :middle rd-want-m1
 :high (+ rd-want-m1 2)))
(if rd-range-m1
 (def rd-m1-d (select rd-range-m1 19)))
(rd-score-update 1 rd-m1-d)
(send rd-dynamic :show-window)
(1d-scroll
 :prompt "2ml =" :location (list 310 500)
 :display rd-range-m1 :choice rd-m1-d
 :action #'(lambda (x) (def rd-m1-d x)
 (rd-score-update 1 x)))
(let*
 ((msg (format nil
 "Do you want to change the search range?")))
 (yes (send modal-button-proto :new "Yes"
 :action #'(lambda () (setf change 1))))
 (no (send modal-button-proto :new "No"
 :action #'(lambda () (setf change 0))))
 (change-dialog (send modal-dialog-proto
 :new (list msg (list yes no))
 :title "To Change Search Range"
 :location (list 230 500))))
 (send change-dialog rmodal-dialog)))

;; confirm the choice of parameter
 (setf tmp-2ml (get-l-value :label "2ml ="
 :value rd-m1-d
 :title "log-F(2m1,2) scores"
 :location (list 400 500)))
 (when tmp-2ml
 (setf v-par (/ tmp-2ml 2))
 (def rd-want-m1 tmp-2ml)
 (def rd-want-m2 2)
 (send rd-dynamic :close)
 (rd-rglm :choice 1 :parameter v-par))))

((= choice 2) ;; use log-F(2,2m2) scores
 (cond
 ((= initial-rd 1) ;; before the first run

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(setf tmp-2m2 (get-1-value
  :label "2m2 =":value rd-want-m2
  :title "log-F(2,2m2) scores"
  :location (list 400 500)))

(when tmp-2m2
  (setf v-par (/ tmp-2m2 2))
  (def rd-want-m2 tmp-2m2)
  (def rd-want-m1 2)
  (rd-rglm :choice 2 :parameter v-par)))
  (= initial-rd 0)
  (rd-setup-search :choice 2)
  (do
    ((change 1))
    ((equal change 0))
    (def rd-range-m2 (sequence-selection
      :string "2M2"
      :low (max .5 (- rd-want-m2 2))
      :middle rd-want-m2
      :high (+ rd-want-m2 2)))
    (if rd-range-m2
      (def rd-m2-d (select rd-range-m2 19)))
      (rd-score-update 2 rd-m2-d)
      (send rd-dynamic :show-window)
      (id-scroll
        :prompt "2m2 ="
        :location (list 310 500)
        :display rd-range-m2
        :choice rd-m2-d
        :action '#'(lambda (x)
          (def rd-m2-d x)
          (rd-score-update 2 x))))
    (let*
      ((mesg (format nil
        "Do you want to change the search range?")))
      (yes (send modal-button-proto :new "Yes"
        :action '#'(lambda () (setf change 1))))
      (no (send modal-button-proto
        :new "No"
        :action '#'(lambda ()
          (setf change 0))))
      (change-dialog (send modal-dialog-proto
        :new (list mesg (list yes no))))
      ..
(defun rd-rglm (#key (choice 0) (parameter 2))
"Run RGLM with dnost and Log Paretos scores."
(cond
  ((= choice 0) ;; log-F(2,2) scores
   (setf f (open "RGLM.IN" :direction :output))
   (format f "title \"trial\"%filexy xy.dat%ryes%dnost\"")
   (close f)
   (system "/home/faculty/rglm/rglm/bin/rglm")
)

  ((= choice 1) ;; log-F(2m1,2) scores
   (setf f (open "RGLM.IN" :direction :output))
   (format f "title \"trial\"%filexy xy.dat%ryes%score lp2 %f%dnst\" parameter)
   (close f)
   (system "/home/faculty/rglm/rglm/bin/rglm")
)

  ((= choice 2) ;; log-F(2,2m2) scores
   (setf f (open "RGLM.IN" :direction :output))
   (format f "title \"trial\"%filexy xy.dat%ryes%score lpl %f%dnost\" parameter)
   (close f)
   (system "/home/faculty/rglm/rglm/bin/rglm")
))

(defun rd-score-update (choice x)
"update quantiles for score search."
(cond
  ((= choice 1)
   (def rd-quant (rearrange-list
                   (logf-quant
                    (ppoints (car (last rd-n-all))) x 2)
                    (order (order rd-resid))))
   (send rd-dynamic :variable-label 0
           (format nil "Log-F (\(\cdot,1f,2\)" rd-m1-d)))
  (;; choice 2)
   (def rd-quant (rearrange-list
                   (logf-quant
                    (ppoints (car (last rd-n-all))) 2 x)
                    (order (order rd-resid))))
   (send rd-dynamic :variable-label 0
           (format nil "Log-F (2,\(\cdot,1f\)" rd-m2-d)))
   (send rd-dynamic :clear nil)
   (send rd-dynamic :add-points rd-quant rd-resid)
   (send rd-dynamic :range 0
           (min rd-quant) (max rd-quant))
   (send rd-dynamic :range 1
           (min rd-resid) (max rd-resid))
   (send rd-dynamic :redraw)
   (send rd-dynamic :add-2-bp rd-quant rd-resid))

(defun rd-setup-search (&key (choice 0))
"Setup the environment for score-selection."

(cond
  ((= choice 1)
   (def rd-m1-d rd-want-m1)
   (setf var-lab (list
                (format nil "Log-F (\(\cdot,1f,2\)" rd-m1-d)
                (format nil "Log-F (\(\cdot,1f,2\)" rd-want-m1))))
  (;; choice 2)
   (def rd-m2-d rd-want-m2)
   (setf var-lab (list
                (format nil "Log-F (2,\(\cdot,1f\)" rd-m2-d)
                (format nil "Log-F (2,\(\cdot,1f\)" rd-want-m2))))
   (def rd-quant (rearrange-list
                   (logf-quant
                    (ppoints (car (last rd-n-all)))
                    rd-want-m1 rd-want-m2))

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(order (order rd-resid)))
(def rd-dynamic (plot-points rd-quant rd-resid
:title "Search"
:variable-labels var-lab
:location (list 250 50) ;; (Left,Top)
:size (list 350 350)
:show nil))
(send rd-dynamic :range 0
(min rd-quant) (max rd-quant))
(send rd-dynamic :range 1
(min rd-resid) (max rd-resid))
(send rd-dynamic :x-axis t t 4)
(send rd-dynamic :y-axis t t 4)
(send rd-dynamic :redraw)
(send rd-dynamic :add-2-bp rd-quant rd-resid)
(require "mei-fun")
(provide "compare-call")

(defun compare-call ()
"Comparing the R- and GR-fits"

;; initialization
(def initial 1)
(def comp-run 0) ; index for which "bded" run
; 0=the first; 1=the second
(def group-b-pt nil)

;; one time event
(get-input-weight)

(def cfits-plot (make-array (list num-gr 2)))
(def cfits-ref-plot (make-array (list num-gr 2)))
(def data-plot (make-array (list num-gr 2)))
(def data-ref-plot (make-array (list num-gr 2)))
(def rd-plot (repeat nil num-gr))
(def et-plot (repeat nil num-gr))
; run comparison program, and obtain information
; from output and weights
(system "/home/faculty/rglm/Bded/April/bded")
(get-output) ;; output of "bded"
(def initial 0)

(plot-tdb) ; prepare plot TDBETAS vs. GR-exp

;; display the comparison dialog
(let*
((prompt "GR-fit with exponent = ")
(list-menu (repeat "" (+ num-gr 2)))
(dummy (setf (select list-menu num-gr)
"Show current accumulative plots"))
(dummy
(setf (select list-menu (1+ num-gr)) "Done"))
(dummy (do-times (i num-gr)
(setf (select list-menu i)
(concatenate 'string prompt
(select gr-exp-str i))))))
(compare-menu (send choose-item-dialog-proto :new)
"Which GR-fit to compare with the R-fit?"
list-menu
:location (list 250 10)
:title "Choice of Comparisons"
:initial 0
:show nil)))

; do action according to the choice of comparison
(do
  ((comp-choice 0))
  ((equal comp-choice (1+ num-gr)) (out-comp))
  (unless (send compare-menu :allocated-p)
    (send compare-menu :allocate))
  (send tdb-plot :show-window)
  (def gr-choice
    (send compare-menu :modal-dialog))
  (setf comp-choice gr-choice)
  (send tdb-plot :hide-window)
;; define actions here
  (when (and comp-choice
    (< comp-choice (1+ num-gr)))
    (cond
      ((= comp-choice num-gr) (show-accumulative))
      ((< comp-choice num-gr)
        (if (= comp-run 1) (def comp-run 0));;reset
          (setf tmp-tdb
            (select (select tdbetas comp-run)
              comp-choice))
          (setf tmp-tbench (select tdb-bench comp-run))
          (if (< tmp-tdb tmp-tbench)
            (mei-message-dialog
              (format nil
"TDBETAS(",2f) is less than its benchmark(",2f)."
              tmp-tdb tmp-tbench)
              :location (list 250 350)))
            (perform-comp))))))

(defun get-input-weight ()
"Get content of initial INPUT and WTSIN file."

;;;;; INPUT ;;;;;
(def original-data (read-data-r "INPUT"))
(def original-n
  (select (select original-data 0) 0))
(def original-p
  (select (select original-data 0) 1))
(def original-p1 (1+ original-p))
(def original-xy (array-to-nested-list
  (transpose ;; all rows and col=1
  (select (select original-data 1)
  (iseq original-n) (iseq 1 original-p))))))
(def xy-lab (get-var-lab))
(def original-index (iseq original-n))
(def original-lab
  (number-to-string original-index))
(def comp-case-len (list original-n))
(def comp-case-index (list
  (copy-list original-index)))
(def comp-case-lab (list
  (copy-list original-lab)))

;;;;; WTSIN ;;;;
(with-open-file (f "WTSIN" :direction :input)
  ;;; read and discard for those NOT-needed info
  (dotimes (i 7) (read f))
  (def num-gr (read f))
  (setf tmp-exp (repeat nil num-gr))
  (dotimes (i num-gr)
    ;;; read and discard NOT-needed info
    (dotimes (j 3) (read f))
    ;;; weight info, actually is exponent
    (setf (select tmp-exp i) (read f))
    ;;; read and discard NOT needed info
    (dotimes (j 2) (read f))))
  (setf tmp-str (number-to-string tmp-exp))
  (def gr-exp (copy-list tmp-exp))
  (def gr-exp-str (copy-list tmp-str)))

(defun get-var-lab ()
  "use dialogs to get variable labels==>tmp-xy-lab."
  ;;; Note the first column is the constant 1
  ;;; in the INPUT file, and we don't need to give
  ;;; it a var-lab
  (let*
((prompt "What is the name of variable ")
(prompt-list (repeat "" original-p))
(tmp-xy-lab (repeat "" original-p))
(lab (number-to-string (iseq 1 original-p)))
(default (repeat "" original-p)) ;default-lab
(dotimes (i original-p)
(if (< i (1- original-p)) ;;; for x-lab
(setf (select default i)
(concatenate 'string "X" (select lab i)))
(setf (select default i) "Y") ;; for y-lab
(setf (select prompt-list i)
(concatenate 'string prompt
(select lab i) "?"))
(setf (select tmp-xy-lab i)
(get-string-dialog (select prompt-list i)
:initial (select default i)
:title "To set up label of variables"))

tmp-xy-lab))

(defun get-output ()
"Get CURRENT files of fort.98 (TDBETAS), and
fort.93 (CFITS)."

; note the p in benchmark is (p - 1) in RGLM.IN
(cond
((= initial 1)
(def tdbetas (list (read-data-file "fort.98")))))
(def tdb-bench (list
(/ (* 4 (~ original-p 2))
(car (last comp-case-len))))
(def cfits (list
(read-data-columns "fort.93" num-gr)))))
(def cfits-bench (list
(* 2 (sqrt
(/ original-p (car (last comp-case-len)))))))

((= initial 0)
(def tdbetas (append tdbetas
(list (read-data-file "fort.98")))))
(def tdb-bench (append tdb-bench (list
(/ (* 4 (~ original-p 2))
(car (last comp-case-len))))))
(def cfits (append cfits (list

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(read-data-columns "fort.93" num-gr)))
(def cfits-bench (append cfits-bench (list
  (* 2 (sqrt
    (/ original-p
      (car (last comp-case-len)))))))))

(defun plot-tdb ()
"Plot GR-exp vs. TDBETAS."
(def tdb-plot (plot-points
gr-exp (select tdbetas comp-run)
:title
  (format nil "TDBETAS (~,2f)"
    (select tdb-bench comp-run))
:variable-labels (list "GR-exponent" "TDBETAS")
:location (list 650 10)
:size (list 250 200)
:show nil))
(send tdb-plot :abline
  (select tdb-bench comp-run) 0)
(send tdb-plot :abline
  (* -1 (select tdb-bench comp-run)) 0)
(send tdb-plot :y-axis t t 4))

(defun perform-comp ()
"Perform actions for each comparison of R- and
GR-fit."

(plot-cfits-data)
(setf del-yn (comp-out-del)) ;; out-deletion

(when (= del-yn 1) ;; after deletion then re-run
  (mei-message-dialog
    "Now we are ready to re-run the comparison
    program with chosen points deleted?"
    :location (list 220 600))
  (def comp-run 1)
  (system "/home/faculty/rglm/Bded/April/bded")
  (get-output) ;; output of "bded"
  (setf tmp-tdb
    (select (select tdbetas comp-run) gr-choice))
  (setf tmp-tbench (select tdb-bench comp-run))
  (if (< tmp-tdb tmp-tbench)
(mei-message-dialog (format nil
"TDBETAS(~,2f) is less than its benchmark(~,2f)."
  tmp-tdb tmp-tbench)
  :location (list 250 350)))
(plot-cfits-data :run 1)
(setf del-yn (comp-out-del))
(get-rdfffits)))

(defun plot-cfits-data (&key (run comp-run))
  "This function will plot cfits and data."

(unless (aref cfits-plot gr-choice run)
  (setf (aref cfits-plot gr-choice run)
    (plot-points
     (select comp-case-index run)
     (select (select cfits run) gr-choice)
     :title
     (format nil "CFITS (+/- ~,2f) (GR-exp=~,2f)"
      (select cfits-bench run)
      (select gr-exp-str gr-choice))
     :variable-labels (list "Case Number" "CFITS")
     :point-labels (select comp-case-lab run)
     :location (list 10 10)
     :size (list 350 350)
     :show nil))
  (send (aref cfits-plot gr-choice run)
     :abline (select cfits-bench run) 0)
  (send (aref cfits-plot gr-choice run)
     :abline (* -1 (select cfits-bench run)) 0)
  (send (aref cfits-plot gr-choice run)
     :linked t)
  (send (aref cfits-plot gr-choice run)
     :showing-labels t)
  (setf (aref cfits-ref-plot gr-choice run)
    (plot-points
     (select comp-case-index run)
     (select (select cfits run) gr-choice)
     :title "Reference Plot for CFITS"
     :variable-labels
     (list "Case Number" "CFITS")
     :point-labels (select comp-case-lab run)
     :location (list 10 450)
(send (aref cfits-ref-plot gr-choice run)
  :abline (select cfits-bench run) 0)
(send (aref cfits-ref-plot gr-choice run)
  :abline (* -1 (select cfits-bench run)) 0)
(send (aref cfits-ref-plot gr-choice run)
  :showing-labels t)
(send (aref cfits-ref-plot gr-choice run)
  :selection (select comp-case-index run))

;;;;;; data plot ;;;;;;
(def xy-data (array-to-nested-list (transpose
  (select (select original-data 1)
    (string-to-number (select comp-case-lab run))
    (iseq 1 original-p)))))
(cond
  ((<= original-p 3)
    (setf (aref data-plot gr-choice run)
      (plot-points xy-data
        :title "Plot of Data"
        :variable-labels xy-lab
        :point-labels (select comp-case-lab run)
        :location (list 380 10)
        :size (list 350 350)
        :show nil)))
  (setf (aref data-ref-plot gr-choice run)
    (plot-points xy-data
      :title "Reference Plot for Data"
      :variable-labels xy-lab
      :point-labels (select comp-case-lab run)
      :location (list 380 450)
      :size (list 350 350)
      :show nil)))
  ((> original-p 3)
    (setf (aref data-plot gr-choice run)
      (scatterplot-matrix xy-data
        :title "Plot of Data"
        :variable-labels xy-lab
        :point-labels (select comp-case-lab run)
        :location (list 380 10)
        :size (list 350 350))
        :show nil)))
(scatterplot-matrix xy-data
:title "Reference Plot for Data"
:variable-labels xy-lab
:point-labels (select comp-case-lab run)
:location (list 380 450)
:size (list 350 350)
:show nil)))

(send (aref data-plot gr-choice run) :linked t)
(send (aref data-plot gr-choice run) :showing-labels t)
(send (aref data-ref-plot gr-choice run) :showing-labels t)
(send (aref data-ref-plot gr-choice run) :selection (select comp-case-index run)))

(send (aref cfits-plot gr-choice run) :selection nil)
(send (aref cfits-plot gr-choice run) :show-window)
(send (aref cfits-ref-plot gr-choice run) :show-window)
(send (aref data-plot gr-choice run) :selection nil)
(send (aref data-plot gr-choice run) :show-window)
(send (aref data-ref-plot gr-choice run) :show-window)
(send (aref data-ref-plot gr-choice run) :show-window)
(if (> original-p 3) (mei-message-dialog "If you want rotation plot of the data, do:
1. use a separate window to run program
2. enter XLS by typing xlispsstat
3. type (load \"comp-rot\")
4. enter variable labels
5. choose three variables to use
6. select points to highlight
7. click Pitch, Roll, or Yaw to rotate"
 :location (list 400 400)))

(defun comp-out-del ()
"give instruction, select points, confirm
selection, delete points, update INPUT, update
comp-case-len, -index, and -lab. Also return
a value indicating delete or not 1=yes, 0=no."

;;; give instructions to choose the "outliers"
(mei-message-dialog
 "To choose outlier, please:
1. use the reference plot below to learn
point labels
2. to select points, eg: points 1 2 3,
type in \"(list 1 2 3)\"
3. CLICK \"Cancel\" button to select NO point
4. points selected will be confirmed."
 :location (list 740 250))

;;; select points
(setf comp-del-tmp (get-value-dialog
 "Expression for indices to select:
 :location (list 740 300)))

;;; confirm selection NOTE: need match
(send (aref cfits-plot gr-choice comp-run)
 :selection (match (car comp-del-tmp)
 (string-to-number
  (select comp-case-lab comp-run))))
(send (aref data-plot gr-choice comp-run)
 :selection (match (car comp-del-tmp)
 (string-to-number
  (select comp-case-lab comp-run))))
(setf comp-del 0)
(mei-yn-dialog
 :mesg "Do you really want to delete those points?"
 :action-yes #'(lambda () (setf comp-del 1))
 :action-no #'(lambda () (setf comp-del 0))
 :title "To confirm choice of points"
 :location (list 740 300))

(send (aref cfits-plot gr-choice comp-run)
 :hide-window)
(send (aref cfits-ref-plot gr-choice comp-run)
 :hide-window)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(send (aref data-plot gr-choice comp-run) :hide-window)
(send (aref data-ref-plot gr-choice comp-run) :hide-window)

;;; perform deletion if yes
(when (= comp-del 1)
  (def group-b-pt (append group-b-pt (car comp-del-tmp)))
  (setf cp-n-tmp
       (- original-n (length group-b-pt)))
  (setf want-row (remove-sublist original-index group-b-pt))
  (setf cp-xy-tmp (select (select original-data 1)
                           want-row (iseq original-p1)))
  (def comp-case-len (append comp-case-len (list cp-n-tmp)))
  (def comp-case-index (append comp-case-index (list (iseq cp-n-tmp))))
  (def comp-case-lab (append comp-case-lab (list (number-to-string want-row))))
  (write-data-R (list (list cp-n-tmp original-p) cp-xy-tmp) "INPUT"))

;;; return index for deletion, yes=1, no=0 comp-del)

(defun show-accumulative ()
  "Show accumulative plots of CFITS, data, RDFITS, and R-EXT."

(let ((ac-menu (send choose-item-dialog-proto :new "Which accumulative plot?"
                                             (list "CFITS and data" "RDFITS and R-EXT"
                                                   "None")
                                             :location (list 250 100)
                                             :title "Accumulative Plots"
                                             :initial 0)))
  (do ((ac-choice 0))
(equal ac-choice 2) ()
(unless (send ac-menu :allocated-p)
  (send ac-menu :allocate))
(setf ac-choice
  (send ac-menu :modal-dialog))
(when (and ac-choice (< ac-choice 2))
  (choose-ac-plot ac-choice)))))

(defun choose-ac-plot (index)
  "Do the choosing of accumulative plot."

(let*
  ((prompt "GR-fit with exponent = ")
   (list-menu (repeat "" (+ num-gr 1)))
   (dummy (setf (select list-menu num-gr) "None"))
   (dummy (dotimes (i num-gr)
     (setf (select list-menu i)
       (concatenate 'string prompt
         (select gr-exp-str i))))))
  (tmp-title (list "Plot of CFITS and data"
    "Plot of RDIFFITS and R-EXT")
  (sub-menu (send choose-item-dialog-proto
    :new "Which Comparison?" list-menu
    :location (list 250 100)
    :title (select tmp-title index)
    :initial 0
    :show nil)))
  (do
    ((sub-choice 0))
    ((equal sub-choice num-gr) ()
     (unless (send sub-menu :allocated-p)
       (send sub-menu :allocate)))
    (setf sub-choice
      (send sub-menu :modal-dialog))
  (cond
    ((= index 0) ;; for CFITS and data plots
      when
      (and sub-choice (< sub-choice num-gr)
       (aref cfits-plot sub-choice 0))
      (send (aref cfits-plot sub-choice 0)
        :selection nil)
      (send (aref cfits-plot sub-choice 0)
(send (aref cfits-ref-plot sub-choice 0) :show-window)
(send (aref data-plot sub-choice 0) :show-window)
(send (aref cfits-ref-plot sub-choice 0) :show-window)
(send (aref data-plot sub-choice 0) :show-window)
(send (aref data-ref-plot sub-choice 0) :show-window)
(mei-message-dialog "See the plot?"
 :location (list 750 250))
(when (aref cfits-plot sub-choice 1)
 (id-scroll
 :action #'(lambda (x) (setf tmp-run x)
 (send (aref cfits-plot sub-choice tmp-run) :show-window)
 (send (aref cfits-ref-plot sub-choice tmp-run) :show-window)
 (send (aref data-plot sub-choice tmp-run) :show-window)
 (send (aref data-ref-plot sub-choice tmp-run) :show-window)
 (send (aref data-ref-plot sub-choice tmp-run) :show-window) :
 :location (list 770 230)
 :display (list 0 1)
 :choice 0)
(send (aref cfits-plot sub-choice 1) :hide-window)
(send (aref cfits-ref-plot sub-choice 1) :hide-window)
(send (aref data-plot sub-choice 1) :hide-window)
(send (aref data-ref-plot sub-choice 1) :hide-window)
(send (aref data-ref-plot sub-choice 1) :hide-window) ;; close out the 2nd run
(send (aref cfits-plot sub-choice 0) :hide-window)
(send (aref cfits-ref-plot sub-choice 0) :hide-window)
(send (aref data-plot sub-choice 0) :hide-window)
(send (aref data-ref-plot sub-choice 0) :hide-window)
(send (aref data-ref-plot sub-choice 0) :hide-window)
(send (aref data-plot sub-choice 0) :hide-window)
(send (aref data-ref-plot sub-choice 0) :hide-window)
(hide-window)) ; close out the 1st run
((= index 1) ;; for RDFFITS/R-EXT
  (when (and sub-choice (< sub-choice num-gr)
    (select rd-plot sub-choice))
    (send (select rd-plot sub-choice)
      :show-window)
    (mei-message-dialog "see the plot?"
      :location (list 750 250))
    (send (select rd-plot sub-choice)
      :hide-window))))

(defun get-rdffits ()
"To get R-diagnosics, RDFFITS and R-EXT."

(setf comp-rd 0)
(ime-yn-dialog
  :mesg "Do you want RDFFITS and R-EXT?"
  :action-yes #'(lambda () (setf comp-rd 1))
  :action-no #'(lambda () (setf comp-rd 0))
  :location (list 100 100))
(when (= comp-rd 1)
  (ime-message-dialog
    "We are now ready to run RGLM one-at-a-time to get RDFFITS and R-EXT. When you see dialog for confirmation of RDFITS statistics, please click YES button every time"
    :location (list 200 350))
  (setf gb-len (length group-b-pt))
  (setf group-a (select (select original-data 1)
    (remove-sublist original-index group-b-pt)
    (iseq original-p1)))
  (def compare-rd (repeat nil gb-len))
  (def compare-et (repeat nil gb-len))

(setf i 0)
(setf run 1)
(setf used-n (+ (- original-n gb-len) 1))
(loop (if (or (= i gb-len) (= run 0)) (return))
  (write-data-R (list
    (list used-n original-p) group-a) "xy.dat")
  (setf f (open "xy.dat" :direction :output
    :if-exists :append))
(dotimes (j original-p1)
  (format f "-%
    (select (select original-data 1)
      (select group-b-pt i) j)))
(format f "-%")
(close f)
(setf run (rglm-rdffits))
(when (= run 1)
  (setf tmp (read-data-columns "RDREAD.OUT" 6))
  (setf (select compare-et i)
    (select (last (select tmp 2)) 0))
  (setf (select compare-rd i)
    (select (last (select tmp 3)) 0)))
(setf i (1+ i)))  ;; the end of the loop

;;; plot RDIFFITS and R-EXT
(when compare-rd
  (setf (select rd-plot gr-choice)
    (plot-points group-b-pt compare-rd
      :title (format nil "RDIFFITS (+/- 2f)"
        (* 2 (sqrt (/ (- original-p 1) used-n))))
      :variable-labels (list "case number" "RDIFFITS")
      :location (list 90 100)
      :size (list 300 300)))
  (send (select rd-plot gr-choice) :y-axis t t 4)
  (send (select rd-plot gr-choice) :abline
    (* 2 (sqrt (/ (- original-p 1) used-n))) 0)
  (send (select rd-plot gr-choice) :abline
    (* -2 (sqrt (/ (- original-p 1) used-n))) 0)

(setf (select et-plot gr-choice)
  (plot-points group-b-pt compare-et
    :title "R-EXT (+/- 2)"
    :variable-labels (list "case number" "R-EXT")
    :location (list 420 100)
    :size (list 300 300)))
  (send (select et-plot gr-choice) :y-axis t t 4)
  (send (select et-plot gr-choice) :abline 2 0)
  (send (select et-plot gr-choice) :abline -2 0)

(meimessage-dialog
  "Plots for RDIFFITS and R-EXT"
(defun out-comp ()
  "Exit from the comparison program"

  (write-data-r original-data "INPUT")
  (dotimes (i num-gr)
    (dotimes (j 2)
      (when (aref cfits-plot i j)
        (send (aref cfits-plot i j) :close)
        (send (aref cfits-ref-plot i j) :close)
        (send (aref data-plot i j) :close)
        (send (aref data-ref-plot i j) :close))))

  (dotimes (i num-gr)
    (when (select rd-plot i)
      (send (select rd-plot i) :close)
      (send (select et-plot i) :close)))

  (format t " \"Done with Comparisons\" ")
REFERENCES


McKean, J. W., Thomas, V. J., and Sievers, G. L. (1989), "A Robust Two-
Stage Multiple Comparison Procedure with Application to a Random Drug Screen," *Biometrics*, 45, 1281–1297.


