TIBCO Spotfire S+® 8.1
Robust Library
User's Guide

November 2008

TIBCO Software Inc.

The correct bibliographic reference for this document is:

Spotfire S+ 8.1 Robust Library User's Guide, TIBCO Software Inc.

Printed in the United States.

Acknowledgments

The Insightful development team for the **Robust Library** consisted of Kjell Konis, Doug Martin, Matias Salibian-Barrera and Jeff Wang. We are grateful to have received very substantial help and guidance from Alfio Marazzi and Victor Yohai as primary consultants, as well as from consultants Ricardo Maronna, Ruben Zamar and David Rocke. Peter Rousseeuw kindly provided a high quality software version of his Fast MCD robust covariance matrix estimate. Alex Randriamiharisoa provided some critical software development help.

This Robust Library would not have been possible without substantial use of software resulting from Alfio Marazzi's comprehensive development of numerical algorithms for robust statistics, as materialized in Marazzi (1993).

Peter Rousseeuw's earlier contributions of LMS and LTS regression and MVE robust covariance matrix software for S-PLUS were instrumental in getting S-PLUS "on the map" with robust methods. Though not focal points of the Robust Library, these methods are still available in Spotfire S+.

This release contains the following contributed S-PLUS code: robust spline functions from Elvezio Ronchetti and Eva Cantoni; robust Cp functions from Robert Staudte and Suzanne Sommer; regression quantiles from Roger Koenker.

The Research and Development of the **Robust Library** was partially supported by NIH Small Business Innovative Research (SBIR) Phase I and II grants 1 R43 GM54450-01A1 and 5 R44 GM54450-03, "Usable Robust Statistical Modeling and Inference" under the direction of R. Douglas Martin as Principal Investigator.

CONTENTS

INTRODUCTION TO ROBUST LIBRARY

1

OUR OVERALL GOAL

Our overall goal is to provide a broad range of robust methods for statistical modeling with the following features:

- **Automatic Computation of both Classical and Robust Estimates** When using the graphical user interface dialog for the Robust Library, the default choice is to automatically compute both the classical and the robust estimate. A special "fit.models" function for fitting multiple models is provided to facilitate computing both classical and robust estimates at the command line.
- **Outlier Data Mining and Comparison Plots**. Diagnostic plots are provided as a fundamental data mining tool that will assist you in quickly identifying outliers, in isolation or in small clusters, and determining whether or not outliers have substantial influence on the classical estimate.
- **Trellis Graphics Diagnostic Comparison Plots**. Trellis graphics are used to display side by side diagnostic plots for comparing classical maximum likelihood estimate (MLE) model fits with robust fits.
- **Robust Statistical Inference**. Robust t-statistics, p-values, F-tests, and bias-detection tests are provided, based on robust covariances and normal distribution approximations for parameter estimates.
- **Robust vs. Classical Inference Comparison is Facilitated**. Pairwise tabular displays of the robust and classical inference results facilitate quick comparison of inference results.
- **Special Scalable Methods for Linear Model Fits and Covariance Matrix Estimation**. Special methods are provided for robust fitting of linear models with large numbers of numeric predictor variables and/or many factor variables with possibly many levels, and for robust covariance matrix estimation with large numbers of variabels and large numbers of observations.

BASIC NOTIONS OF ROBUSTNESS

Classical maximum likelihood estimates (MLE) based on assumed idealized distributions almost always lack robustness toward outliers in the sense that outliers can have a very substantial influence on maximum likelihood parameter estimates. This is true not only of Gaussian maximum likelihood estimates such as the least squares estimates of linear models and the classical covariance matrix estimates, but also of a variety of non-Gaussian maximum likelihood estimates such as the MLE's for the parameters of exponential, Weibull and gamma distributions.

The probability distribution models that generate outliers are often close to the assumed ideal distribution in the central portion of the distribution, but differ from the ideal distribution in the tails of the distribution in a seemingly small but potent manner. The major consequence of such outlier-generating distributions is that the maximum likelihood parameter estimates based on the ideal distribution can suffer from large *bias* and substantially increased *variability* (or equivalently decreased statistical *efficiency*). Furthermore, the resulting bias persists even as the sample size n increases toward infinity, while the increased variability typically tends to zero like n^{-1}. Thus control of bias is more important for larger sample sizes.

Robust estimation methods were invented to deal with the above problems, and the important properties of a good robust estimator are as follows:

- In data-oriented terms: parameter estimates and the associated robust model fit are minimally influenced by outliers, and provide a good fit to the bulk of the data.
- Diagnostic plots based on the robust fit will allow you to quickly and easily identify outliers, and determine whether or not outliers are affecting the classical MLE model fits.
- In probability-oriented terms, a robust method minimizes the bias in coefficient estimates due to outlier-generating distribution models, while at the same time achieving a high *efficiency* when the data has the assumed ideal distribution (equivalently, the variance is not much larger than that of the MLE at the assumed ideal distribution)

- The robust parameter estimates provide good approximate statistical inference based on the large sample size approximate normality of the parameter estimates.

For further information, read the sections on Theoretical Details in subsequent chapters.

ROBUST MODELING METHODS

The following robust modeling methods are provided in the **Robust Library**.

- Robust Linear Regression and Model Selection
- Robust ANOVA
- Robust Covariance and Correlation Estimation
- Robust Principal Component Analysis
- Robust Fitting of Poisson and Logistic GLIM's
- Robust Discriminant Analysis
- Robust Parameter Estimates for Asymmetric Distributions

Robust Regression for the Linear Model

Two robust linear model fitting methods are included: (1) An MM-estimate, and (2) a new adaptive estimate due to Gervini and Yohai (1999). The MM-estimate is the default choice. The new adaptive estimate has the feature that it is asymptotically *efficient* when the data is Gaussian, i.e., is as good as least-squares when the data is Gaussian, while at the same time controlling bias due to outliers in a nearly optimal manner.

computation time

Both estimators described above require a highly robust initial estimate, and for the case all the predictor variables are numeric we continue to use a sampling approach to computing an initial S-estimate. It is known that such an approach has exponential complexity of order 2^p where p is the number of predictor variables. We provide some tabled estimates, based on empirical studies, of approximate computation times of the robust linear model fit as function of *p*, the number of observations, and the computer platform. The practical limit on the number of independent variables for reasonably quick computation with present generation workstations is roughly 15. In addition, we print out estimates of the time remaining for a robust fit so that you can decide whether to wait for the result or defer the computation to a more convenient time.

fast robust regression procedure

A fast procedure for obtaining initial estimates is implemented following Pena and Yohai (1999). Although these estimates are not guaranteed to have high breakdown point, they result in enormous speed improvement for large problems. The reliability of these estimates has been confirmed by simulation.

fitting models with both numeric and categorical variables

When you have factor type variables as well as numeric variables, each factor level requires an additional linear model parameter and dummy predictor variable. In such cases you may often find yourself with many more than 15 predictor variables. On the other hand, the predictor variables used to model the factor variables only take on the values 0 and 1, and for such variables a high breakdown point initial S-estimate is not really required. A least absolute deviations (LAD) type M-estimate will suffice. Based on this observation, Maronna and Yohai (1999) designed an alternating S-estimate/M-estimate method for fitting linear models with both factor and numeric predictor variables, which we have implemented for this release. This method will allow you to handle linear models with factor variables that require many more than 15 parameters.

robust model tests and robust model selection

Robust F-tests and robust Wald tests are provided. In addition, a robust model selection criterion called RFPE is provided. RFPE is a robust version of Akaike's Final Prediction Error criterion (FPE). Also, a robust backward elimination method for model selection is provided.

Robust ANOVA

A robust ANOVA capability has been introduced using the same final M-estimate as is used for the MM-estimate for the linear model. The main difference is that since the ANOVA setup has only factor variables, a high-breakdown initial S-estimator is not required and it suffices to use an LAD initial estimate.

Robust Covariance Estimation

We provide the Fast Minimum Covariance Determinant (MCD) covariance matrix estimate of Rousseeuw and van Driessen (1999), the Donoho-Stahel projection-type estimator, an M-estimator, and a new scalable estimator based on pairwise robust covariance estimates due to Maronna and Zamar (2001). The pairwise covariance matrix

estimate is adjusted so that it is positive definite. The default choice is the Donoho-Stahel estimator for sufficiently small numbers of observations and variables, and otherwise is the MCD estimator. Also included are: plots for the comparison of robust and classical covariance estimates: an overlaid eigenvalue scree plot, a distance-distance plot, an ellipses plot for smaller matrices, and an image display for larger covariance matrices.

Robust Principal Component Analysis

The Robust Library includes functions for conducting a principal component analysis (PCA) based on a robust covariance matrix estimate. You can use any of the estimators mentioned above for the robust covariance matrix estimate.

Robust Logistic and Poisson Generalized Linear Models

Robust generalized linear model fitting has been provided for the logistic and Poisson link functions, using the conditionally unbiased bounded-influence function approach of Künsch, Stefanski and Carroll (1989). For logistic regression models, the Robust Library provides two additional robust estimates: a weighted MLE estimate with Mallows-type leverage-dependent weights and a consistent estimate based on Copas's (1988) misclassification model (Carroll and Pederson, 1993).

Deviance residuals qq-plots

Deviances are not normally distributed in GLIM's, and hence normal QQ-plots of deviances can be misleading when assessing the fit of a GLIM model. The QQ-plot of deviances estimates the distribution of the deviances and uses the resulting estimated quantiles to draw the QQ-plot (Garcia Ben and Yohai, 2000).

Robust Discriminant Analysis

A robust discriminant analysis function is provided, based on one of the above robust covariance matrix estimates. Error rate estimates are provided via a Monte Carlo simulation based on the fitted model.

Parameter Estimates for Asymmetric Distributions

The robust library includes functions that calculate the following two classes of estimates for Gamma, Weibull and Lognormal distributions: (i) Truncated-mean estimates, and (ii) optimal bounded-influence M-

estimates. The truncated-mean estimates are simpler to compute than the optimal M-estimates, and are intuitive in nature to most users. Thus the truncated-mean estimates are the default choice.

SPECIAL FEATURES OF THE ROBUST LIBRARY

Plots for Outlier Detection and Comparing Fits

When both a classical and a robust fit are computed, all plots selected on the **Plots** page of the dialog are created in a Trellis display with the classical and robust results displayed sided by side. For linear regression models including ANOVA models, QQ-plots and residuals density estimates are also available as overlaid plots. For regression models, plots of standardized residuals or deviances versus robust distances, introduced by Rousseeuw and von Zomeren (1990) are provided.

Multiple Model Fits and Comparisons Paradigm

A command line creator function `fit.models` is provided for creating an object of class "`fit.models`", along with `print`, `plot` and `summary` methods for this class of objects. You can use the function `fit.models` to create an object that contains both the least-squares and robust fits. Then you can make convenient comparison of the fits with respect to inference results by using `summary`, and convenient visual comparison of the fits and visual outlier detection by using `plot`.

GUI for the NT/ Windows Version

A uniform model-fitting dialog design has been created with the following basic features. Each robust model fitting dialog uses the current dialog for the classical model fitting method, with two minimal changes to accommodate the robust method. The first is that on each dialog's **Model** page you have three fitting method choices: (1) Compute both the classical and the robust fit, (2) Compute only the classical fit, and (3) Compute only the robust fit, with choice (1) being the default. Second, an **Advanced** page containing the various parameters and tuning constants used in the numerical procedures for the robust methods has been added to each dialog. Most users will not want to bother with these options.

DATA SETS IN THE ROBUST LIBRARY

The following data sets are included in the **Robust Library** and can be used to carry out the examples in this document and to experiment with the library.

- `stack.dat` This data set has been analyzed by a large number of statisticians. See Dodge (1996) for the history of this data set.
- `wagner.dat` The data set has been analyzed by Wagner (1994), Hubert and Rousseeuw (1997), and Maronna and Yohai (1999).
- `breslow.dat` This data set is used by Breslow (1996).
- `lawson.dat` This data set is from Lawson and Gold (1988).
- `milkcomp.dat` This data set is analyzed by Atkinson.
- `bushfire.dat` This data set is analyzed by Maronna and Yohai (1995).
- `woodmod.dat` This data set is a modified version of the wood gravity data from Rousseeuw and Leroy (1987).
- `hawkins.dat` This data set is used by Hawkins, Bradu and Kass (1984).
- `sim.dat` This artificial data set is used for illustrating robust F tests and robust model fitting for robust linear regression.

In addition, the S+PLUS built-in data set **oilcity** is used for illustrating the control of Advanced options in robust modeling.

In order to conveniently view and use these data sets in the Windows version of Spotfire S+, follow the instructions for viewing the Robust Library data sets in the next section.

LOADING THE ROBUST LIBRARY

Loading the library from the NT/ Windows GUI

To load the Robust Library from the NT/Windows GUI, select **File ► Load Library...** from the S+PLUS menu bar to bring up the following dialog window.

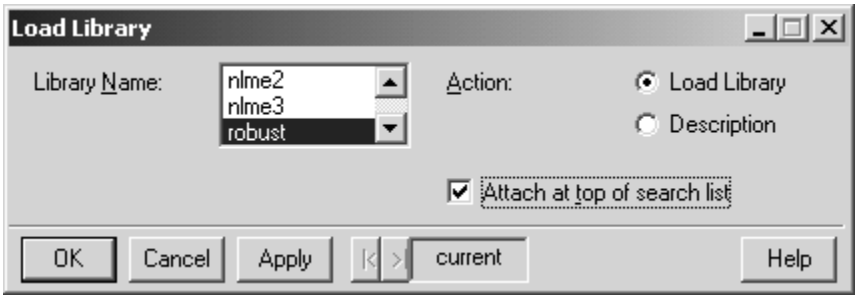


Figure 1.1: *Load Library Dialog*

Select Robust from the list of library names, and make sure to check Attach at top of search list. Click OK to load the library. After the Robust Library is loaded, a menu will be added to the S+PLUS menubar. Most of the functions provided by the Robust Library can be accessed through this menu.

Viewing the Robust Library Data Sets

If you use the Object Explorer you will want to be able to view and use the example data sets included in the **Robust Library**. To do this, right click (after loading the library) on the **Data** icon and select Advanced from the context menu to open the Database Filter dialog.

1

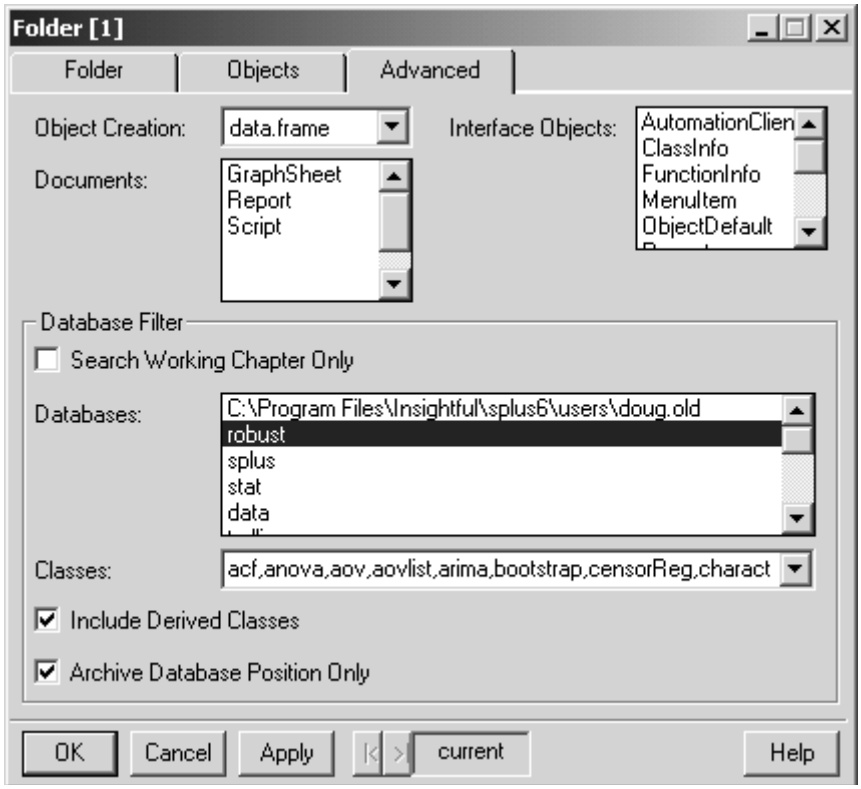


Figure 1.2: *Database Filter Dialog*

Uncheck the Search Working Chapter Only check box. Select Robust from the list of databases and enter `data.frame` Classes field (or select data.frame from the Classes drop-down list box). Then click OK. Notice that shortcuts to the **Robust Library** data sets have been added in the right pane of the Object Explorer.

Alternatively, in the Object Explorer you can expand the SearchPath object, then expand the robust object to access the data sets in the **Robust Library** directly.

Loading the Library from the Command Line

Use the following command to load the library from the commands window:

```
> library(robust, first=T)
```

This command will attach the **Robust Library** in position 2 and add the robust menu to the Spotfire S+ menubar.

ROBUST LINEAR REGRESSION

2

OVERVIEW OF THE METHOD: A SPECIAL M-ESTIMATE

You are fitting a general linear model of the form

$$y_i = x_i^T \beta + \varepsilon_i, \; i = 1, \; ..., \; n$$

with p-dimensional independent predictor (independent) variables x_i and coefficients β, and scalar response (dependent) variable y_i. Spotfire S+ computes a robust M-estimate $\hat{\beta}$ which minimizes the objective function

$$\sum_{i=1}^{n} \rho\left(\frac{y_i - x_i^T \beta}{\hat{s}}\right)$$

where $\hat{s}$ is a robust scale estimate for the residuals and ρ is a particular optimal symmetric *bounded* loss function, described in the section Theoretical Details. Alternatively $\hat{\beta}$ is a solution of the estimating equation

$$\sum_{i=1}^{n} x_i \psi\left(\frac{y_i - x_i^T \hat{\beta}}{\hat{s}}\right) = 0$$

where $\psi = \rho'$ is a redescending (nonmonotonic) function. The shapes of the ρ and $\psi = \rho'$ functions are shown in Figure 2.17.

The above minimization problem can have more than one local minima, and correspondingly the estimating equation above can have multiple solutions. Spotfire S+ deals with this by computing special highly robust initial estimates $\hat{\beta}$ and $\hat{s}$, using the methods described in the section Theoretical Details. Then Spotfire S+ computes the final estimate $\hat{\beta}$ as the local minimum of the M-estimate objective function nearest to the initial estimate. We refer to an M-estimate of this type and computed in this special way as an MM-estimate, a term introduced by Yohai (1987).

COMPUTING LS AND ROBUST FITS WITH THE WINDOWS GUI

Computing Both LS and Robust Fits

You easily obtain both a least squares and robust linear model fit for the so called "stack loss" data using the Robust Linear Regression dialog in the **Robust Library**. The stack loss data is known to contain highly influential outliers, and is included in the **Robust Library** as the data frame `stack.dat`. Display the **Robust Library** data sets in the left-hand pane of Spotfire S+ Object Explorer using one of the methods recommended in the Introduction chapter and select `stack.dat`. The right-hand pane of the Object Explorer displays the four variables in `stack.dat`: the dependent (response) variable `Loss`, and the three independent (predictor) variables `Air.flow`, `Water.Temp` and `Acid.Conc`. First select the response variable `Loss`, and then

)

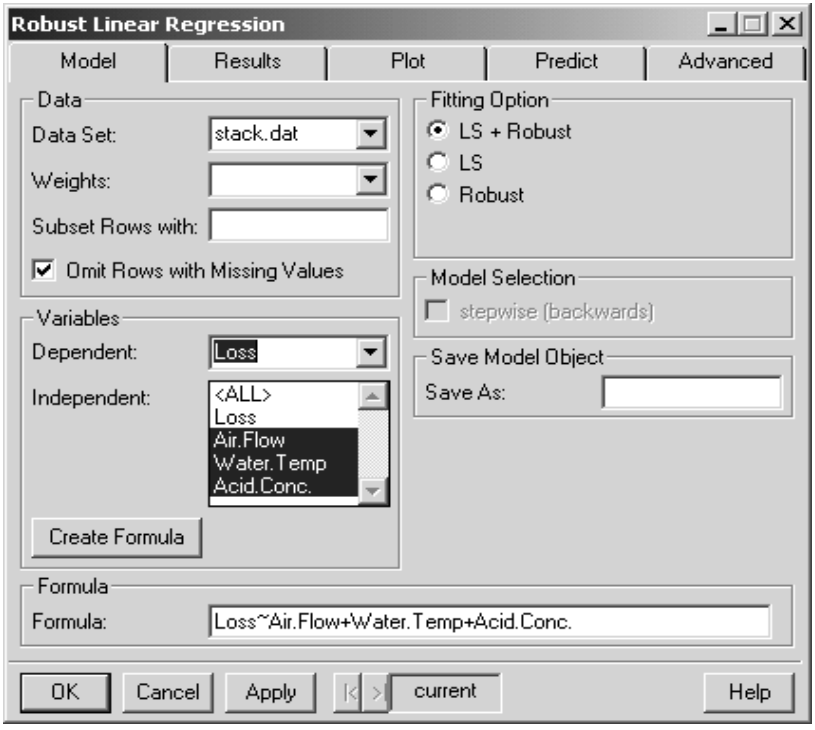


Figure 2.1: *The Linear Regression Dialog: Model Page*

select the three independent (predictor) variables (you can do this by shift clicking on `Acid.Conc`. Choose **Robust ► Linear Regression** from the menubar. The dialog shown in Figure 2.1 appears. Because you selected the response variable `Loss` first, followed by the three predictor variables, the **Formula** field is automatically filled in with correct formula `Loss ~ Air.Flow + Water.Temp + Acid.Conc.` for modeling Loss in terms of the three predictor variables.

Note that the **Model** page of this dialog looks exactly like that of the Linear Regression dialog in Spotfire S+, except for the **Fitting Options** choices, with the default choice **LS + Robust** (both least squares and robust fits are computed) and alternate choices **LS** (least squares fit only) and **Robust** (robust fit only) and the **Advanced** tab. Click on the **Advanced** tab to access optional advanced features of the robust fitting method. These are discussed in the section Advanced Options For Robust Regression, and we suggest you wait until reading that section to experiment with the robust fitting method options.

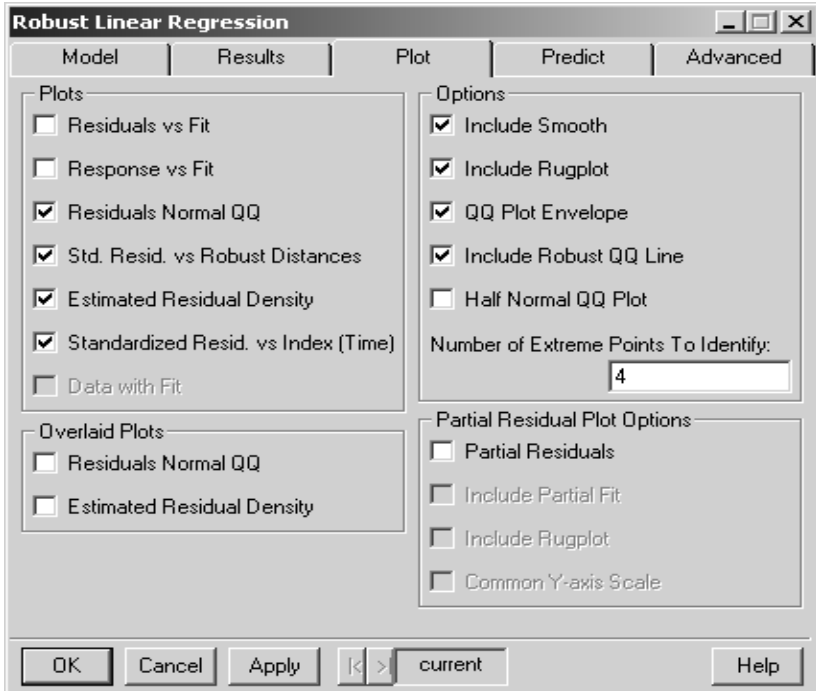


Figure 2.2: *The Linear Regression Dialog: Plot Page*

Click on the tabs labeled **Results**, **Plot** and **Predict** to look at those dialog pages. You will notice that the **Results** and **Predict** pages are identical to those of the **Linear Regression** dialog in Spotfire S+. However, the **Plot** page shown in Figure 2.2 is different in that it has several new **Plots** region entries: *Std. Resid. vs. Robust Distances*, *Estimated Residual Density*, *Standardized Resid. vs. Index (Time)* and *Data with Fit*. The latter is greyed out when there is more than one independent variable. The **Plot** page also has a new **Overlaid Plots** region with the entries: *Residuals Normal QQ* and *Estimated Residual Density*. The latter are only available when you have chosen the default choice **LS + Robust** on the **Model** page.

We have made the default choices of plots indicated by the checked boxes. This will encourage you to quickly compare the LS and robust versions of these plots and quickly determine whether or not there are any outliers in the data, and whether or not the outliers have an impact on the least squares fit. In the **Number of Extreme Points to Identify** text box, replace the 3 by 4.

Click **OK** to compute both the LS and robust fits, along with the four diagnostic comparison plots and other standard statistical summary information. The results appear in a **Report** window and four tabbed pages of a **Graph Sheet**, respectively.

The Diagnostic Plots

Each of the **Graph Sheet** pages contains a Trellis display for the LS and robust fit, as shown below.

Normal QQ-Plots of Residuals

As seen in Figure 2.3, the normal QQ-plot for the LS fit residuals shows at most one outlier, while the one for the robust fit reveals four outliers. The outliers are those points that fall outside the 95% simulation envelopes for the normal qq-plot, shown as dotted lines. This reveals one of the most important advantages of a good robust fit relative to a least squares fit: the least squares fit is highly influenced by outliers in such a way that the outliers are not clearly revealed in the residuals, while the robust fit clearly exposes the outliers.

You also note that if you ignore the outliers, a normal distribution is a pretty good model for the residuals in both cases. However, the slope of the central linear portion of the normal QQ-plot of the residuals for the robust fit is noticeably smaller than that for the LS fit. This indicates that the normal distribution fit to the robust residuals,

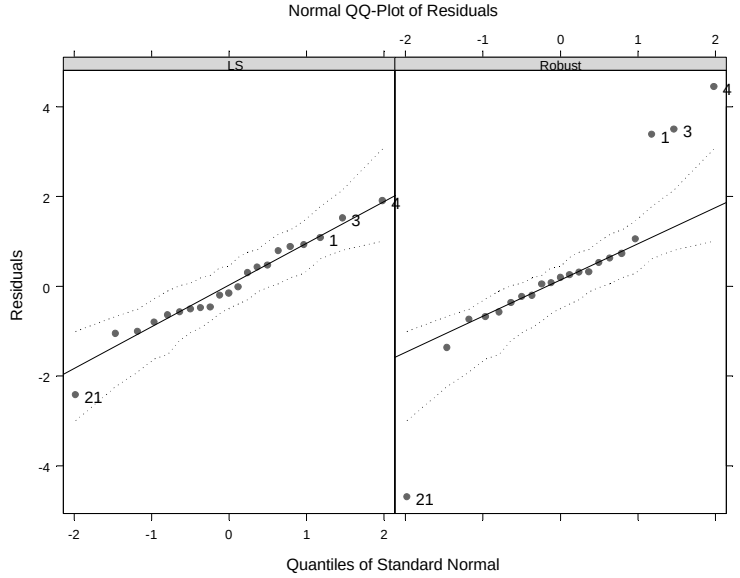


Figure 2.3: *LS and Robust Normal QQ-Plots of Residuals: stack.dat*

ignoring the outliers, has a substantially smaller standard deviation than the normal distribution fit to the LS residuals. In this sense, the robust method provides a better fit to the bulk of the data.

Probability Density Estimates of Residuals

Figure 2.4 displays the (kernel) probability density estimates for the residuals for the least squares and robust fits, and it clearly reveals the existence of outliers that adversely influence the LS fit. The story here is consistent with that provided by the normal QQ-plot comparisons: you see that density estimate of the LS residuals is much broader in the central region than that of the robust residuals, and is rather skewed and not centered on zero. The density estimate of the residuals for the robust fit is very compact and centered on zero in the central region, and exhibits two distinct bumps that indicate the presence of outliers. From this point of view, the robust fit again provides a better fit to the bulk of the data and indicates the presence of outliers.

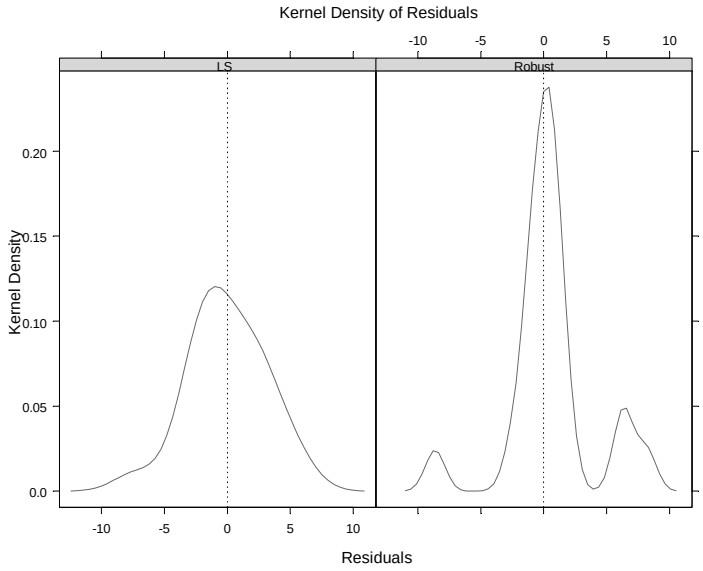


Figure 2.4: *LS and Robust Density Estimates of Residuals: stack.dat*

Standardized Residuals versus Robust Distances

A highly useful plot of scaled residuals versus robust distances of the predictor variables was invented by Rousseeuw and van Zomeren (1990). For both the LS and robust fits, the *robust distances* are the Mahalanobis distances based on a robust covariance matrix estimate for the predictor variables, as described in the section Theoretical Details. A large robust distance for a predictor variable indicates that the predictor variable has *leverage* that might exert undue influence on the fit. The scaled residuals for the LS fit are the residuals divided by the standard error of the residuals. The scaled residuals for the robust fit are the residuals divided by a robust scale estimate for the residuals, obtained as part of the robust fitting method.

The standardized residuals vs. robust distances plots for both the least squares and robust fits are shown in Figure 2.5. Following Rousseeuw and van Zomeren (1990), the horizontal dashed lines are located at +2.5 and -2.5, and the vertical line is located at the upper .975 percent point of a chi-squared distribution with p degrees of freedom, where p

= 3 in this case. Points outside the horizontal lines are regarded as residual outliers, and points to the right of the vertical line are regarded as leverage points or *x-outliers*.

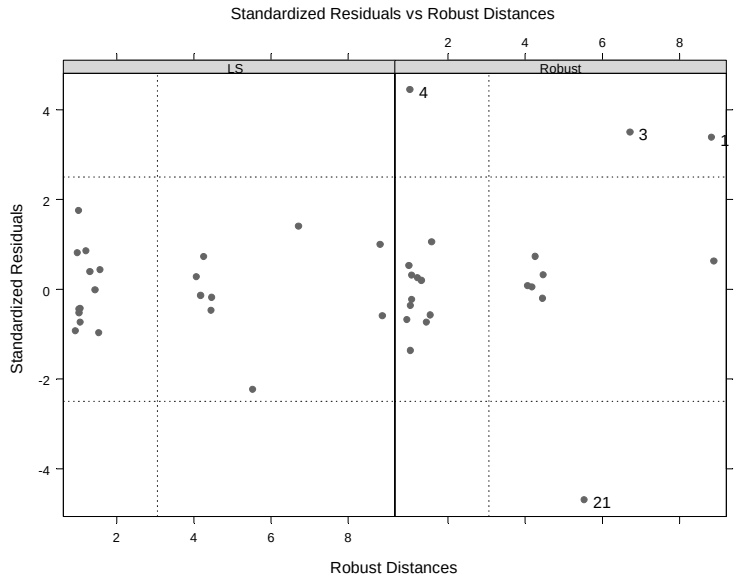


Figure 2.5: *LS and Robust Standardized Residuals vs. Robust Distances: stack.dat*

In this case the LS fit produces no residuals outliers and four x-outliers, whereas the robust fit produces four residuals outliers and four x-outliers. Three of the four x-outliers for the robust fit are also residuals outliers, while one x-outlier is not a residuals outlier. The interpretation is that three of the x-outliers have substantial influence on the LS fit, while the fourth x-outlier does not. The robust fit is not much influenced by outliers, whether they occur in the response space or the predictor space, or both.

This example illustrates the problem of outlier *masking* in least squares fits, i.e., the influence of outliers on the least squares parameter estimates distorts the parameter estimates in such a manner that the outliers can not be detected in plots of the LS residuals. The robust estimate does not suffer from this problem.

Standardized Residuals versus Index (Time)

Figure 2.6 shows the standardized residuals vs. index (time) plots for both the LS and robust fits. As in the previous plot, the standardized residuals for the LS fit are the residuals divided by the standard error of the residuals, and the standardized residuals for the robust fit are the residuals divided by a robust scale estimate for the residuals. If the response variable is a time series, then this plot is a time series plot.

From Figure 2.6, you can see that the LS fit does not reveal any outlier, while the robust fit again clearly reveals the four outliers in the data set: the first three occur at the startup of the underlying chemical process and the other one in the end when the process is shut down.

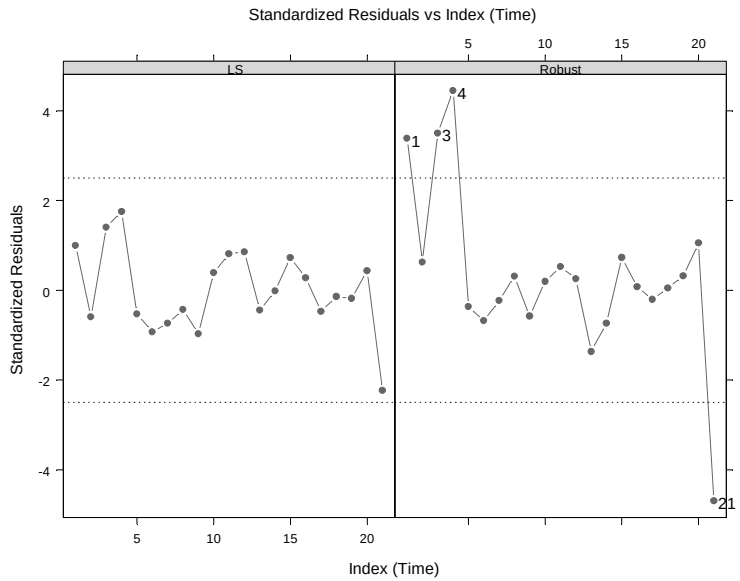


Figure 2.6: *LS and Robust Standardized Residuals vs. Index (Time): stack.dat*

The Statistics Report

The **Report** window contains the following results.

```
   *** Classical and Robust Linear Regression ***

Calls:
Robust    lmRob(formula = Loss ~ Air.Flow + Water.Temp +
Acid.Conc., data = stack.dat, na.action = na.exclude)
```

```
    LS    lm(formula = Loss ~ Air.Flow + Water.Temp +
Acid.Conc., data = stack.dat, na.action = na.exclude)

Residual Statistics:
           Min      1Q  Median      3Q     Max
Robust -8.6299 -0.6713  0.3594  1.1507  8.1740
    LS -7.2377 -1.7117 -0.4551  2.3614  5.6978

Coefficients:
                      Value Std. Error  t value Pr(>|t|)
Robust (Intercept) -37.6525   5.0026    -7.5266   0.0000
    LS (Intercept) -39.9197  11.8960    -3.3557   0.0038
Robust    Air.Flow   0.7977   0.0713    11.1886   0.0000
    LS    Air.Flow   0.7156   0.1349     5.3066   0.0001
Robust  Water.Temp   0.5773   0.1755     3.2905   0.0043
    LS  Water.Temp   1.2953   0.3680     3.5196   0.0026
Robust  Acid.Conc.  -0.0671   0.0651    -1.0297   0.3176
    LS  Acid.Conc.  -0.1521   0.1563    -0.9733   0.3440

Residual Scale Estimates:
: 1.837 on 17 degrees of freedom
: 3.243 on 17 degrees of freedom

Proportion of variation in response(s) explained by
model(s):
Robust : 0.6205
    LS : 0.9136

Bias Tests for Robust Models:

Robust:
Test for Bias:
            Statistics P-value
 M-estimate       2.75    0.60
LS-estimate       2.64    0.62
```

The standard errors, the t-statistics, and the p-values of the robust coefficient estimates for the robust fit are themselves robust because they are computed using a robust covariance matrix for the

parameter estimates. The *Proportion of variation in response explained by model*, or multiple R^2, for the robust fit is a robust version of the classical least-squares R^2.

There is also a *Test for Bias* in the summary statistics provided in the **Report** window. This provides two statistical tests of the bias: the first for bias of the final M-estimate relative to a highly robust initial estimate, and the second for the bias of the LS estimate relative to the final M-estimate. In this example, the p-values for these tests are .60 and .62, indicating that for both comparisons there is little evidence of bias.

Read the section Theoretical Details to find out how these robust inference quantities are computed.

COMPUTING LS AND ROBUST FITS AT THE COMMAND LINE

Computing Both LS and Robust Fits

If you prefer to work at the Spotfire S+ command line, you can use the `fit.models` function in the **Robust Library** to compute both an LS and a robust linear model fit and store them as a single S-PLUS object, say `stack.fits`:

```
> stack.fits <- fit.models(list(Robust = "lmRob",
+ LS = "lm"), Loss ~. , data = stack.dat)
```

Now view a brief summary of the results:

```
> stack.fits

Calls:
Robust    lmRob(formula = Loss ~ ., data = stack.dat)
    LS    lm(formula = Loss ~ ., data = stack.dat)

Coefficients:
               Robust       LS
(Intercept) -37.6525 -39.9197
   Air.Flow   0.7977   0.7156
 Water.Temp   0.5773   1.2953
 Acid.Conc.  -0.0671  -0.1521
```

```
Residual Scale Estimates:
Robust : 1.837 on 17 degrees of freedom
    LS : 3.243 on 17 degrees of freedom
```

Use the `summary` function to obtain a more complete summary of the model fitting results:

```
> summary(stack.fits)

Calls:
Robust    lmRob(formula = Loss ~ ., data = stack.dat)
    LS    lm(formula = Loss ~ ., data = stack.dat)

Residual Statistics:
           Min      1Q  Median      3Q     Max
Robust -8.6299 -0.6713  0.3594  1.1507  8.1740
    LS -7.2377 -1.7117 -0.4551  2.3614  5.6978

Coefficients:
                     Value Std. Error  t value Pr(>|t|)
Robust (Intercept) -37.6525   5.0026    -7.5266   0.0000
    LS (Intercept) -39.9197  11.8960    -3.3557   0.0038
Robust    Air.Flow   0.7977   0.0713    11.1886   0.0000
    LS    Air.Flow   0.7156   0.1349     5.3066   0.0001
Robust  Water.Temp   0.5773   0.1755     3.2905   0.0043
    LS  Water.Temp   1.2953   0.3680     3.5196   0.0026
Robust  Acid.Conc.  -0.0671   0.0651    -1.0297   0.3176
    LS  Acid.Conc.  -0.1521   0.1563    -0.9733   0.3440

Residual Scale Estimates:
Robust : 1.837 on 17 degrees of freedom
    LS : 3.243 on 17 degrees of freedom

Proportion of variation in response(s) explained by
model(s):
Robust : 0.6205
    LS : 0.9136
```

```
Correlations:
Robust
            (Intercept) Air.Flow Water.Temp Acid.Conc.
(Intercept)  1.0000
   Air.Flow  0.0049      1.0000
 Water.Temp -0.0828     -0.7077   1.0000
 Acid.Conc. -0.8442     -0.2885  -0.0453     1.0000

    LS
            (Intercept) Air.Flow Water.Temp Acid.Conc.
(Intercept)  1.0000
   Air.Flow  0.1793      1.0000
 Water.Temp -0.1489     -0.7356   1.0000
 Acid.Conc. -0.9016     -0.3389   0.0002     1.0000


Bias Tests for Robust Models:

Robust:
Test for Bias:
            Statistics P-value
 M-estimate       2.75    0.60
LS-estimate       2.64    0.62
```

The Diagnostic Plots

You can also make comparison plots with the `plot` function:

```
> plot(stack.fits)

Make plot selections (or 0 to exit):

1: plot: All
2: plot: Normal QQ-Plot of Residuals
3: plot: Estimated Kernel Density of Residuals
4: plot: Robust Residuals vs Robust Distances
5: plot: Residuals vs Fitted Values
6: plot: Sqrt of abs(Residuals) vs Fitted Values
7: plot: Response vs Fitted Values
8: plot: Standardized Residuals vs Index (Time)
9: plot: Overlaid Normal QQ-Plot of Residuals
10: plot: Overlaid Estimated Density of Residuals
Selection(s): 9, 10
```

Note that in the **Robust Library**, you can select more than one plot from the above menu of choices, which was not possible before. On Windows platform, this will result in a multi-paged **Graph Sheet**, with each plot on one page. In this case, both the overlaid normal QQ-plot and residuals density are shown in Figure 2.8. One thing you might have noticed is that in the overlaid normal QQ-plot, the residuals are plotted on the horizontal axis, instead of on the vertical axis as shown in Figure 2.3 for the Trellis display. This makes it easier to compare the normal QQ-plot with the density plot. In Figure 2.8 you can easily visualize the impacts of outliers in the robust fit.

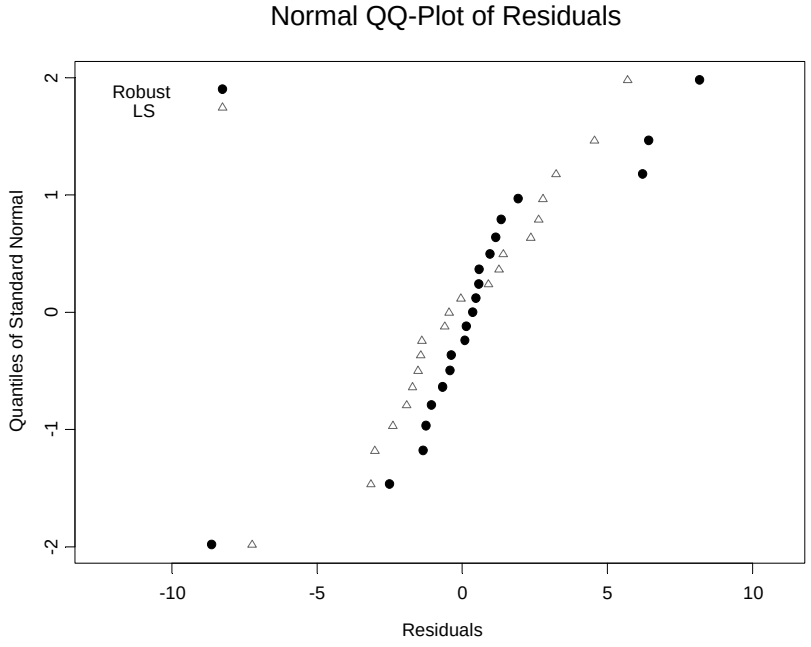


Figure 2.7: *Overlaid Normal QQ-Plot of Residuals*

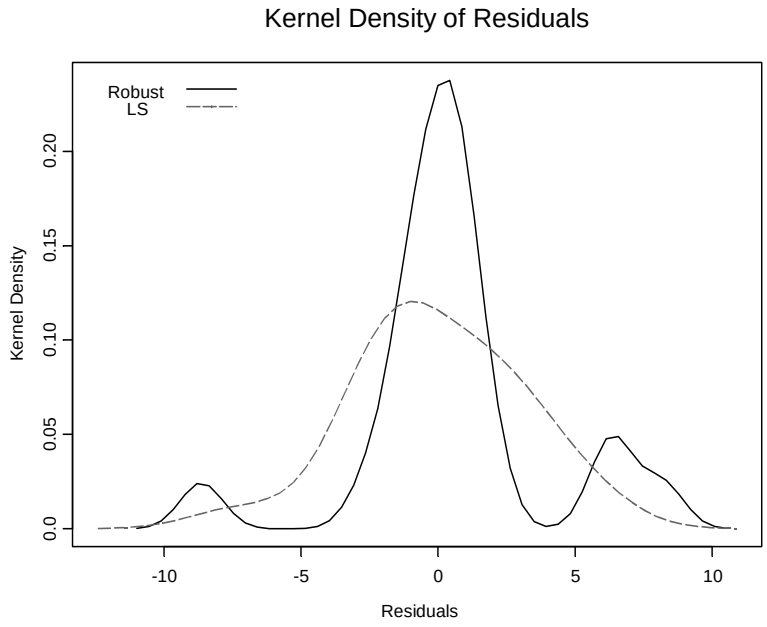


Figure 2.8: *Overlaid Kernel Density of Residuals*

Computing Only a Robust Fit

Use the function `lmRob` to compute only a robust fit:

```
> stack.robfit <- lmRob(Loss ~., data = stack.dat)
> stack.robfit
Call:
lmRob(formula = Loss ~ ., data = stack.dat)

Coefficients:
 (Intercept)  Air.Flow Water.Temp  Acid.Conc.
   -37.65246 0.7976856  0.5773405 -0.06706018

Degrees of freedom: 21 total; 17 residual
Residual scale estimate: 1.837073
```

You can also use the `summary` and `plot` functions to get more extensive summary results and plots, just as in the case of a least squares fit "`lm`" object.

Computation Time Required

For the size of most regression problems, the robust regression method requires a computationally intensive resampling method to obtain an initial robust estimate of the regression parameters and residual scale, as a starting point for computing the final robust estimates. This initial estimate requires a number of samples and corresponding computational time proportional to 2^p, where p is the number of regression parameters. Obviously when p is large, the computation time can quickly become prohibitive.

Recently Pena and Yohai (1999) proposed a fast procedure for obtaining a reliable initial regression estimate, which can be used as a starting point for high efficiency estimates. Although the initial estimates from the fast procedure are not guaranteed to have high breakdown point, Pena and Yohai (1999) showed that they are comparable with other available robust estimates for a wide range of problems.

By default, Spotfire S+ employs the random resampling algorithm for initial estimates when the number of variables is smaller than 15, and switches to the fast procedure when the number of variables is greater than 15.

The tables below compare the computation times required by the resampling algorithm and the fast procedure, as a function of p and the number of observations n, for Spotfire S+ on a Sun SPARC Ultra-60 with 1024MB memory.

Table 2.1: *Spotfire S+ User Time with Random Sampling Initial Estimate Method*

	n = 50	**n = 100**	**n = 150**	**n = 200**	**n = 250**	**n = 300**	**n = 500**
p = 5	0.20	0.21	0.22	0.24	0.25	0.29	0.34
p = 10	0.73	0.90	1.08	1.26	1.49	1.67	2.59
p = 15	28.54	34.82	41.86	49.84	58.01	64.73	92.98
p = 20	1580.47	1812.71	2099.97	2444.63	2663.29	2942.46	4001.77

Table 2.2: *Spotfire S+ User Time with Pena-Yohai Fast Initial Estimate Method*

	n = 50	**n = 100**	**n = 150**	**n = 200**	**n = 250**	**n = 300**	**n = 500**
p = 5	0.20	0.22	0.23	0.26	0.28	0.28	0.36
p = 10	0.28	0.32	0.34	0.37	0.44	0.53	0.58
p = 15	0.36	0.44	0.57	0.53	0.60	0.82	1.03
p = 20	0.51	0.71	0.85	1.11	1.15	1.65	2.10
p = 30	N/A	1.05	1.94	2.08	2.02	2.39	6.34
p = 50	N/A	3.24	4.38	5.52	17.24	14.91	14.18

In the case you choose to use the random resampling initial estimate whe p is greater than 15 by choosing it on the Advanced tab of Robust Linear Regression dialog, Spotfire S+ informs you in the case of long computation time requirements for the robust regression method by printing out estimates of the remaining computing time. The printout occurs in the **Report** window on the NT/Windows platform, as shown in Figure 2.9 (and in the Command Window on a UNIX/ LINUX platform).

NOTE: For small values of n and p, Spotfire S+ automatically does exhaustive sampling for the initial estimate. Specifically, this happens for $n < 250$ for $p = 2$, and for $n < 80$ for $p = 3$. Otherwise, random resampling is used as the default for $p < 16$.

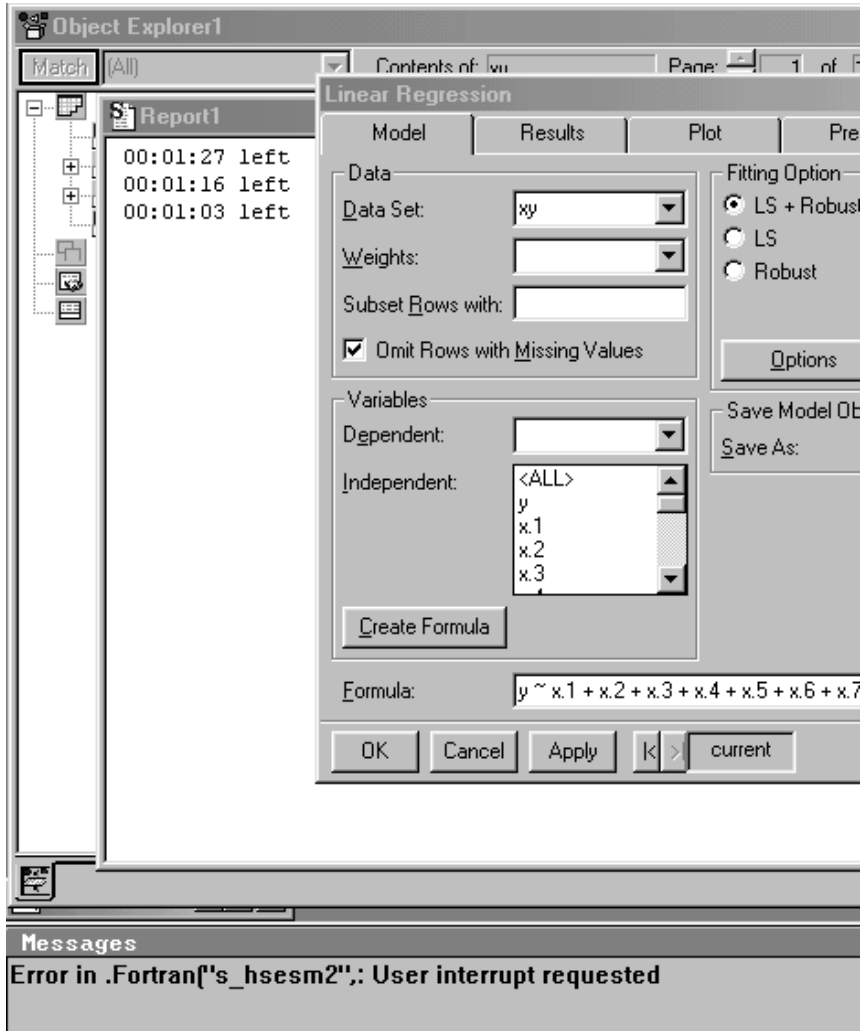


Figure 2.9: *Computation Time Left in Report Window*

The printout of the remaining time estimate using the usual hours:minutes:seconds format. So in the above example, the first printout states that approximately one minute and twenty-seven seconds remain, and the time between successive printouts is approximately twelve seconds.

In the event you want to defer the computation of the robust estimate until another time such as over your coffee break or lunch, press **ESC** once on an Windows version (control-C on a UNIX/LINUX version) and wait until the error message appears in the **Messages** window as shown.

Fitting Models with Both Numeric and Factor Variables

Often you fit linear models with both numeric and factor variables. When the factor variables have many levels, you may be fitting a linear model with considerably more independent variables than fifteen or so. An example of this type is provided by the data set `wagner.dat` included with this library, and a portion of this data set is displayed below. For this data set `y` is the response variable.

Region	PA	GPA	HS	GHS	y	Period
1	46.84	-2.60	1.68	0.20	0.97	1
2	35.54	-1.42	1.67	0.63	2.14	1
3	28.42	-1.48	1.71	0.12	6.13	1
4	32.54	-4.51	1.37	0.32	7.36	1
5	28.92	-0.88	2.14	-0.08	3.63	1
6	36.61	-1.39	3.00	0.45	-4.30	1
7	34.71	-2.22	2.94	0.27	2.06	1
8	24.32	-5.11	3.57	-0.55	-18.64	1
9	35.15	-0.16	3.27	0.03	5.15	1
10	34.06	-3.86	2.74	0.19	6.88	1
11	37.94	-4.61	2.07	0.38	-1.24	1
12	35.88	-2.17	1.57	-0.11	-1.31	1
13	31.28	-1.90	2.74	-0.57	1.73	1
14	33.61	2.02	1.92	0.32	0.44	1
15	33.86	0.75	0.86	0.46	-15.53	1
16	43.24	-4.41	1.82	0.52	-10.99	1
17	42.65	-2.28	1.52	-0.17	0.60	1
18	37.19	-2.75	2.39	0.40	3.71	1
19	49.70	-4.86	1.16	0.09	-2.38	1
20	41.96	-4.59	2.00	-0.12	-1.35	1
21	28.86	-2.11	5.17	0.46	-1.08	1
1	44.24	3.80	1.88	0.13	8.47	2
2	34.12	-3.33	2.30	1.04	2.76	2
3	26.94	-1.71	1.83	1.28	24.08	2
..	.....					

This data set contains four numeric variables: `PA`, `GPA`, `HS`, `GHS`, two factor variables `Region` and `Period`, and response variable `y`. `Region` has twenty-one levels and `Period` has three levels, resulting in a total of $4 + 1 + 20 + 2 = 27$ independent variables.

If you try to fit a linear model using the function `lmRobMM`, available in current versions of Spotfire S+, you will find that it takes a long time (some hours) to compute the fit (or the algorithm fails due to singularity problems). This is because `lmRobMM` uses a subsampling approach with exponential complexity in the number of independent variables, as described earlier.

The **Robust Library** deals with this problem by using a new alternate *S/M-estimate* computing algorithm due to Maronna and Yohai (1999) for robustly fitting linear models which contain factor variables with possibly many levels. Spotfire S+ automatically uses the new S/M-estimate algorithm whenever the linear model data contains at least one factor variable. For further information see the Theoretical Details section Alternating S and M Initial Estimate.

Compute LS and robust fits to `wagner.dat` in a manner similar to the one you used for the stack loss data set `stack.dat`. Open the **Robust Library** data sets in the Object Explorer and select `wagner.dat`. The right-hand pane of the Object Explorer displays the seven variables in `wagner.dat`. First select the dependent (response) variable `y`. Then select the other variables by pressing the CTRL while selecting them (you have to do it this way for `wagner.dat` because the response variable is not the top icon, and so you can not select it first and then SHIFT select the bottom icon to get all the dependent variables as you did with `stack.dat`). Now choose **Robust ► Linear Regression** from the menubar, and proceeding just as you did with the stack loss data set. The resulting residuals diagnostic plot comparisons for the LS and robust fits are shown below.

(Alternatively, you could skip selecting variables in the Object Browser and just select them in the Robust Linear Regression dialog box).

Note that when a linear model contains factor variables as well as continuous (numeric) variables, robust distances are computed only for the continuous variables (it does not make much sense to look at robust distances for factor variables, which are represented by values of zero and one for the corresponding independent variables).

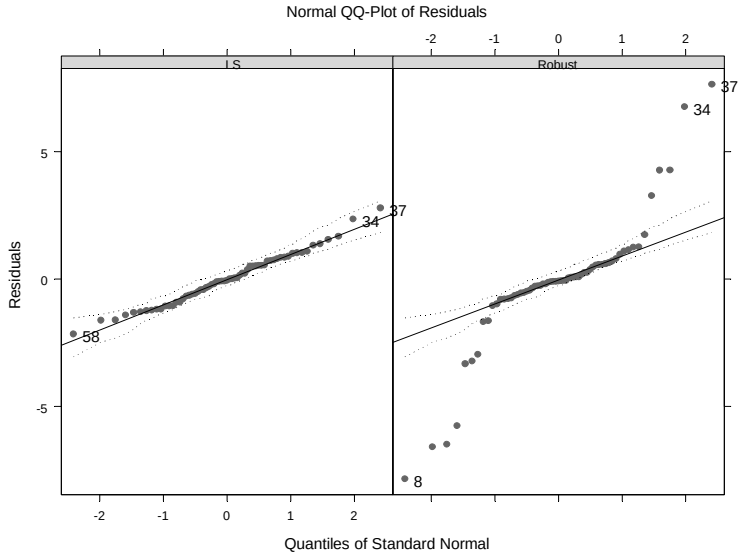


Figure 2.10: *LS and Robust Normal QQ-Plots of Residuals: wagner.dat*

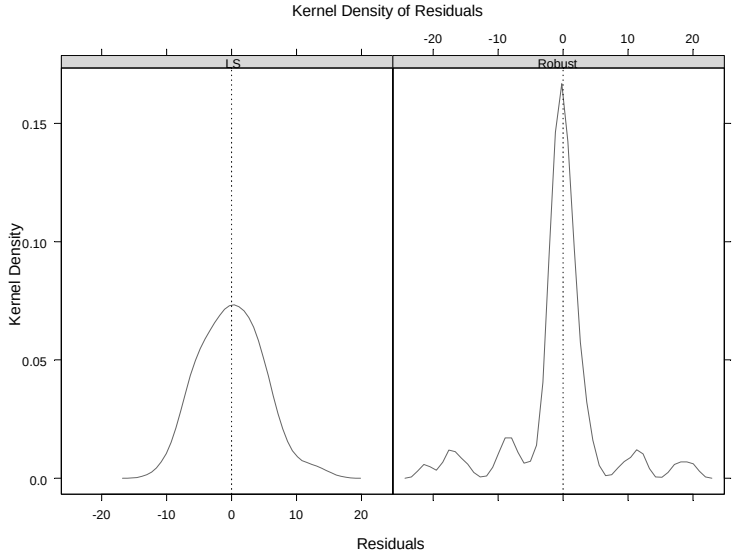


Figure 2.11: *LS and Robust Density Estimates of Residuals: wagner.dat*

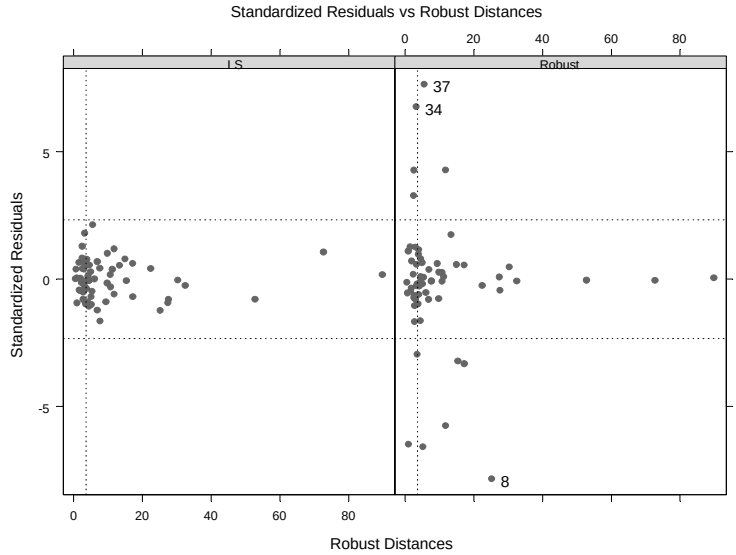


Figure 2.12: *LS and Robust Standardized Residuals vs. Robust Distances: wagner.dat*

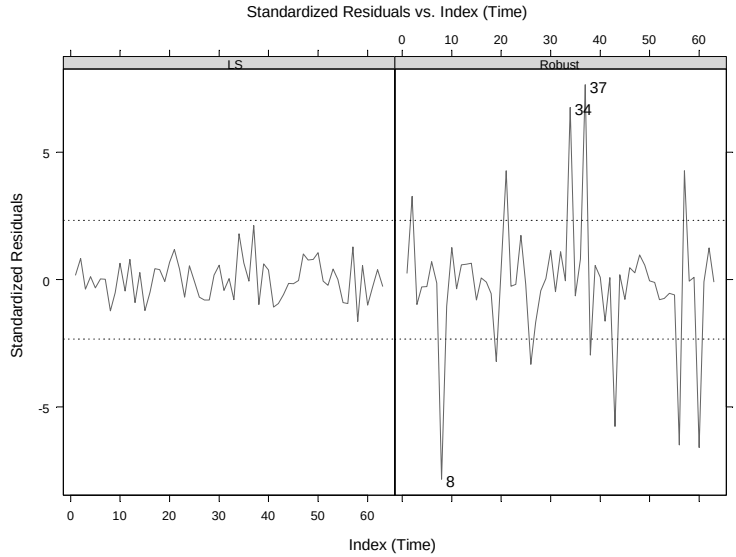


Figure 2.13: *LS and Robust Standardized Residuals vs. Index (Time): wagner.dat*

Figure 2.10-Figure 2.13 show quite dramatic differences between the LS and robust fits: The LS fits give no hints whatsoever of outliers, and the robust method provides a good fit to the bulk of the data and exposes a number of outliers quite clearly.

The **Report** window output for this example is shown below:

```
  *** Classical and Robust Linear Regression ***

Calls:
Robust  : lmRob(formula = y ~ Period + GHS + HS + GPA + PA +
Region, data = wagner.dat, na.action = na.exclude)
    LS  : lm(formula = y ~ Period + GHS + HS + GPA + PA +
Region, data = wagner.dat, na.action = na.exclude)

Residual Statistics:
              Min       1Q   Median       3Q      Max
Robust : -21.0047  -1.6697  -0.2091   1.5360  20.4836
    LS : -10.2302  -3.4272  -0.2351   3.4048  13.2492

Coefficients:
                     Value Std. Error  t value Pr(>|t|)
Robust (Intercept) -40.4171  73.5980    -0.5492   0.5863
    LS (Intercept)   4.7556  23.5957     0.2015   0.8414
Robust     Period1   4.3807   2.6139     1.6759   0.1024
    LS     Period1   1.9477   1.4944     1.3033   0.2007
Robust     Period2   3.9658   1.1962     3.3152   0.0021
    LS     Period2   1.0876   1.0953     0.9930   0.3273
Robust         GHS   2.7902   2.0520     1.3597   0.1824
    LS         GHS   6.1485   1.7159     3.5832   0.0010
Robust          HS   3.7261   5.4735     0.6808   0.5004
    LS          HS   5.0899   2.1070     2.4157   0.0209
Robust         GPA   0.3912   1.3309     0.2939   0.7705
    LS         GPA   0.0737   0.5583     0.1320   0.8958
Robust          PA   1.1309   1.8571     0.6089   0.5464
    LS          PA  -0.4104   0.7058    -0.5815   0.5645
Robust     Region1   2.1205   8.6427     0.2453   0.8076
    LS     Region1  -5.1072   5.5911    -0.9135   0.3671
Robust     Region2   8.9457   7.7568     1.1533   0.2564
    LS     Region2   2.1396   3.5075     0.6100   0.5457
Robust     Region3   3.6278   3.2025     1.1328   0.2648
    LS     Region3   1.2105   1.5928     0.7600   0.4522
```

```
Robust     Region4   1.6062   2.2043    0.7286   0.4709
    LS     Region4  -0.1202   1.1281   -0.1065   0.9158
Robust     Region5  -2.8809   1.9829   -1.4529   0.1549
    LS     Region5  -2.5068   0.8667   -2.8925   0.0064
Robust     Region6  -0.3746   1.7531   -0.2137   0.8320
    LS     Region6  -0.7652   0.7343   -1.0421   0.3043
Robust     Region7   1.3170   1.3252    0.9938   0.3269
    LS     Region7  -2.4701   1.2616   -1.9580   0.0580
Robust     Region8   0.0130   0.7499    0.0173   0.9863
    LS     Region8   0.4503   0.4824    0.9334   0.3568
Robust     Region9  -0.0148   0.4902   -0.0302   0.9761
    LS     Region9  -0.0349   0.4032   -0.0865   0.9316
Robust    Region10  -0.5491   0.5011   -1.0959   0.2804
    LS    Region10   0.2104   0.4260    0.4939   0.6244
Robust    Region11  -0.2878   0.4718   -0.6100   0.5457
    LS    Region11  -0.0978   0.3249   -0.3011   0.7650
Robust    Region12   0.1403   0.7538    0.1861   0.8534
    LS    Region12   0.5725   0.3422    1.6729   0.1030
Robust    Region13  -0.2356   0.6115   -0.3854   0.7022
    LS    Region13  -0.1733   0.3510   -0.4937   0.6245
Robust    Region14  -0.7599   0.5812   -1.3075   0.1993
    LS    Region14  -0.2735   0.3100   -0.8823   0.3835
Robust    Region15  -1.2966   0.5723   -2.2655   0.0296
    LS    Region15  -0.3064   0.3279   -0.9343   0.3564
Robust    Region16  -0.2686   0.5598   -0.4798   0.6343
    LS    Region16   0.3953   0.4052    0.9758   0.3357
Robust    Region17   0.0871   0.5872    0.1483   0.8830
    LS    Region17  -0.0266   0.2529   -0.1051   0.9169
Robust    Region18  -0.2807   0.5688   -0.4936   0.6246
    LS    Region18   0.5132   0.4305    1.1921   0.2410
Robust    Region19  -0.3779   0.1529   -2.4712   0.0183
    LS    Region19  -0.0294   0.2137   -0.1377   0.8912
Robust    Region20  -0.8073   0.7121   -1.1338   0.2644
    LS    Region20  -1.3662   0.4549   -3.0033   0.0048

Residual Scale Estimates:
Robust : 2.68 on 36 degrees of freedom
    LS : 6.229 on 36 degrees of freedom
```

```
Proportion of variation in response(s) explained by
model(s):
Robust : 0.596
    LS : 0.773

Bias Tests for Robust Models:
Robust:
Test for Bias:
            Statistics P-value
 M-estimate      17.17   0.927
LS-estimate       3.41   1.000
```

ROBUST MODEL SELECTION

It is not enough for you to robustly fit a linear model when you are trying to decide which of two linear models with different independent/predictor variables to use, or which of several alternative linear models with different sets of independent/predictor variables to use. You also need *robust test statistics* and *robust model selection criteria*. In this section we first briefly mention robust t-tests, and then show you how to use Spotfire S_ to obtain robust F-tests and robust Wald tests for determining which of two candidate models is preferred. After that we show you how to use a new robust model selection criterion called Robust Final Prediction Error (RFPE) criterion for selecting a "best" model from a set of several candidate models.

Since we have not yet implemented a dialog access to use the robust tests and RFPE, we show you how to use them at the command line.

Robust F Tests

The data set `sim.dat` is a simulated data set provided in the **Robust Library**. The data was created by first generating four independent standard normal random variables, x1, x2, x3, x4, and then added outliers in special locations. Then we generated the response y according to the linear model equation:

$$y = b_1 x1 + b_2 x2 + b_3 x3 + b_4 x4 + u$$

where the first two coefficients have value one and the third and fourth coefficients have value zero, and the error term u is normally distributed.

Use the function pairs

```
> pairs(sim.dat)
```

to obtain the pairwise scatter plots of the five-dimensional data consisting of the response and four independent variables. The result is shown in the figure below.

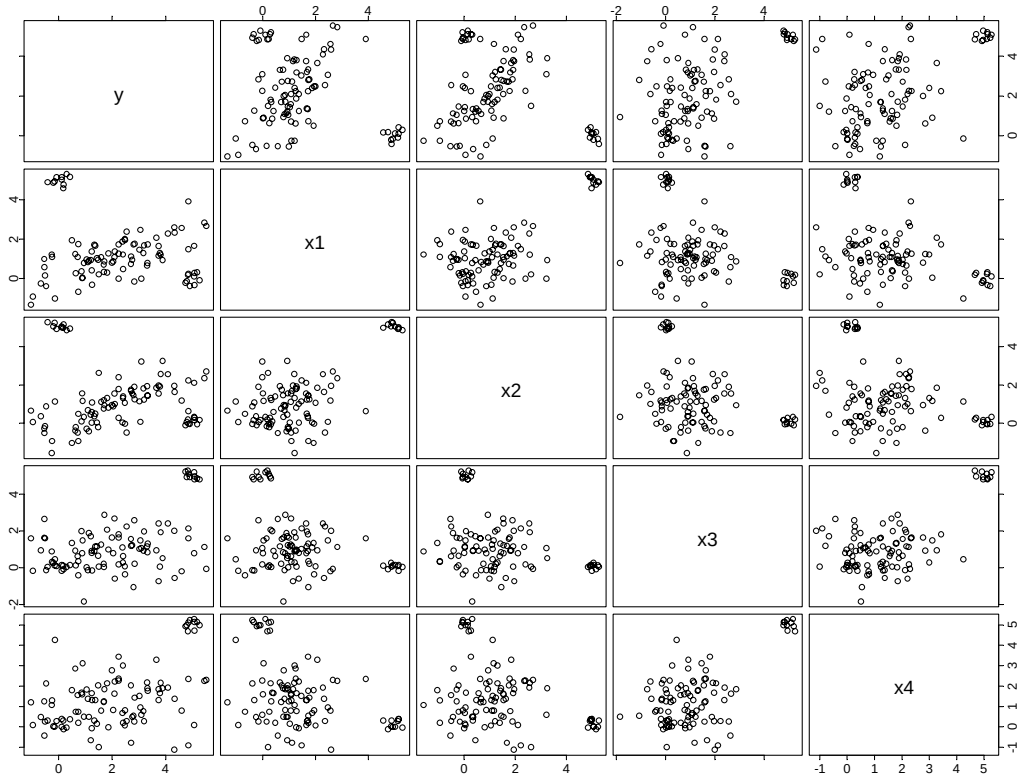


Figure 2.14: *Pairwise Scatter plots of Simulated Linear Model Data with Outliers*

Now make two least squares linear model fits, one with just the first two variables and one with all four variables:

```
> lm.mod12 <- lm(y ~ x1+x2, data = sim.dat)

> lm.mod1234 <- lm(y ~ x1+x2+x3+x4, data = sim.dat)
```

The short summary of the `lm.mod12` fit is as follows:

```
> lm.mod12

Call:
lm(formula = y ~ x1 + x2, data = sim.dat)

Coefficients:
 (Intercept)        x1        x2
    2.368499 -0.175945 0.0217107

Degrees of freedom: 100 total; 97 residual
Residual standard error: 1.801764
```

The coefficients are nowhere near their common true values of one. If you use the `summary` function on `lm.mod12`, you will find that these coefficient estimates are not significantly different than zero.

The short summary of the `lm.mod1234` fit is:

```
> lm.mod1234

Call:
lm(formula = y ~ x1 + x2 + x3 + x4, data = sim.dat)

Coefficients:
 (Intercept)         x1        x2        x3        x4
   0.8810634 0.05003338 0.1185325 0.4257607 0.3557092

Degrees of freedom: 100 total; 95 residual
Residual standard error: 1.511241
```

The first two coefficients are nowhere near their common true value of one, while the third and fourth coefficients are far from their common true value of zero. If you use the `summary` function on `lm.mod1234`, you will find that the first two coefficients are not significant, while the third and fourth are highly significant. These results are quite opposite of the truth.

Now use the `anova` function to compute a classical F-test of whether or not the third and fourth coefficients belong in the model:

```
> anova(lm.mod12, lm.mod1234)

 Analysis of Variance Table

Response:
        Terms ResDf   RSS    Test   Df    SS     F Value
1     x1 + x2   97  314.8961
2 x1+x2+x3+x4   95  216.9657 +x3+x4  2  97.93038 21.43976

          Pr(F)
1
2 2.068392e-008
```

The (classical) F-test erroneously tells you that the third and fourth variables should be in the model!

Now make two robust model fits, the first using the first two variables x1 and x2, and the second using all four variables x1, x2, x3, x4.

```
> rob.mod12 <- lmRob(y ~ x1+x2, data = sim.dat)

> rob.mod1234 <- lmRob(y ~ x1+x2+x3+x4, data = sim.dat)
```

The short summaries of these two robustly fitted models are as follows:

```
> rob.mod12

Call:
lmRob(formula = y ~ x1 + x2, data = sim.dat)

Coefficients:
 (Intercept)       x1       x2
 -0.04584343 1.072822 1.015621

Degrees of freedom: 100 total; 97 residual
Residual scale estimate: 0.7566563
```

```
> rob.mod1234

Call:
lmRob(formula = y ~ ., data = sim.dat)

Coefficients:
 (Intercept)       x1       x2           x3          x4
 -0.02257299 1.072947 1.017503 -0.005768044 -0.01629436

Degrees of freedom: 100 total; 95 residual
Residual scale estimate: 0.7776201
```

You notice that `rob.mod12` provides coefficient estimates that are quite close to the true values of one. You also notice that `rob.mod1234` provides estimates for the first two coefficients that are quite close to their true values of one, and estimates for the third and fourth coefficients that are quite close to their true values of zero.

Now use the `anova` function on these two robustly fitted models to compute a robust F test:

```
> anova(rob.mod12,rob.mod1234)

Response: y
              Terms     Df    RobustF     P(>RobustF)
1           x1 + x2
2 x1 + x2 + x3 + x4      2 0.02121456      0.9890456
```

The test accepts the null hypothesis that the third and fourth coefficients are not significant.

Robust Wald Tests

The default test used by `anova` is a robust F test. You can also use anova to compute a robust Wald test based on robust estimates of the coefficients and covariance matrix. To use the robust Wald test, use the optional argument `test = "RWald"`:

```
> anova(rob.mod12,rob.mod1234, test = "RWald")
Response: y
              Terms     Df       Wald  P(>Wald)
1           x1 + x2
2 x1 + x2 + x3 + x4      2 0.07997027 0.9608037
```

which gives a result quite similar to that of the robust F test.

Robust FPE for Model Selection

Although many robust estimators have been constructed in the past, the issue of robust model selection has not received its due attention. For robust model selection, Spotfire S+ provides a Robust Final Prediction Error (RFPE) criterion proposed by V. Yohai (1997). This criterion is a robust analogue to the Akaike's Final Prediction Error (FPE) criterion.

You may use the RFPE criterion to choose a best model when robustly fitting linear models from the Robust Linear Regression dialog by selecting the Robust fitting option and checking the stepwise (backwards) check box, as shown in the figure 2.15 below.

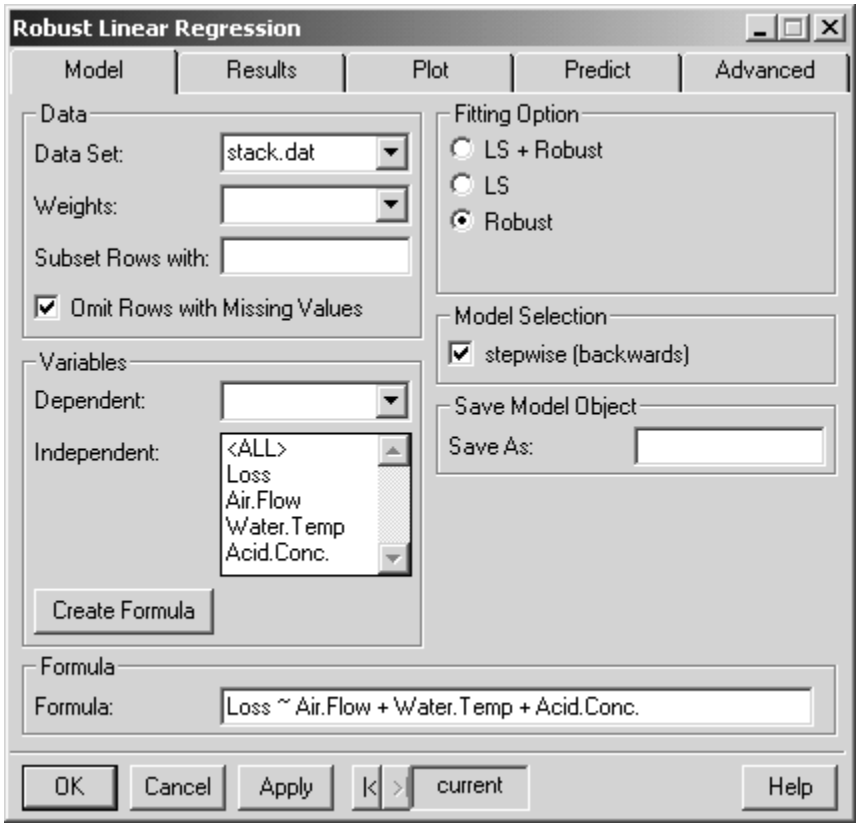


Figure 2.15:

When considering a variety of model choices with respect to several different choices of predictor variables, you choose the model with the smallest value of RFPE. See the section Theoretical Details for further information.

The RFPE criterion is used by the function `step` to compute a backward stepwise robust method of model selection (only *backward* stepwise robust regression is available in this release of the **Robust Library**). For example, when you use `step` on the robustly fitted model rob.mod1234:

```
> step(rob.mod1234)

Start:  RFPE= 79.7096
 y ~ x1 + x2 + x3 + x4

Single term deletions

Model:
y ~ x1 + x2 + x3 + x4

scale:  0.7776201

       Df     RFPE
<none>     79.7096
    x1  1 117.1077
    x2  1 143.8299
    x3  1  79.2279
    x4  1  79.2475

Step:  RFPE = 79.2279
 y ~ x1 + x2 + x4


Single term deletions

Model:
y ~ x1 + x2 + x4

scale:  0.7776201
```

```
       Df     RFPE
<none>     79.2279
    x1  1 124.4032
    x2  1 129.2550
    x4  1  78.7647

Step:  RFPE = 78.7647
 y ~ x1 + x2


Single term deletions

Model:
y ~ x1 + x2

scale:  0.7776201

       Df     RFPE
<none>     78.7647
    x1  1 129.0664
    x2  1 160.7258
Call:
lmRob(formula = y ~ x1 + x2, data = sim.dat)

Coefficients:
 (Intercept)       x1       x2
 -0.04584343 1.072822 1.015621

Degrees of freedom: 100 total; 97 residual
Residual scale estimate: 0.7566563
```

Recall that `sim.dat` was generated by a model in which only the two coefficients associated with x1 and x2 were non-zero and have the common value one. RFPE has clearly chosen the correct robustly fitted model!

ADVANCED OPTIONS FOR ROBUST REGRESSION

In this section, you will learn how to change the default settings of some control parameters for the robust estimator so as to obtain particular estimates that fit your purpose. You can change those options either through a GUI dialog for Spotfire S+ or from the command line. From the command line, most of the default settings can be changed through the functions `lmRob.robust.control` and `lmRob.genetic.control`. Only the commonly used control parameters are introduced in this section. For the default settings of other parameters and how to change them, see the online help file for `lmRob.robust.control` and `lmRob.genetic.control`.

Launch the GUI Dialog

Selecting the **Advanced** tab in the dialog shown in Figure 2.1 brings up the following page:

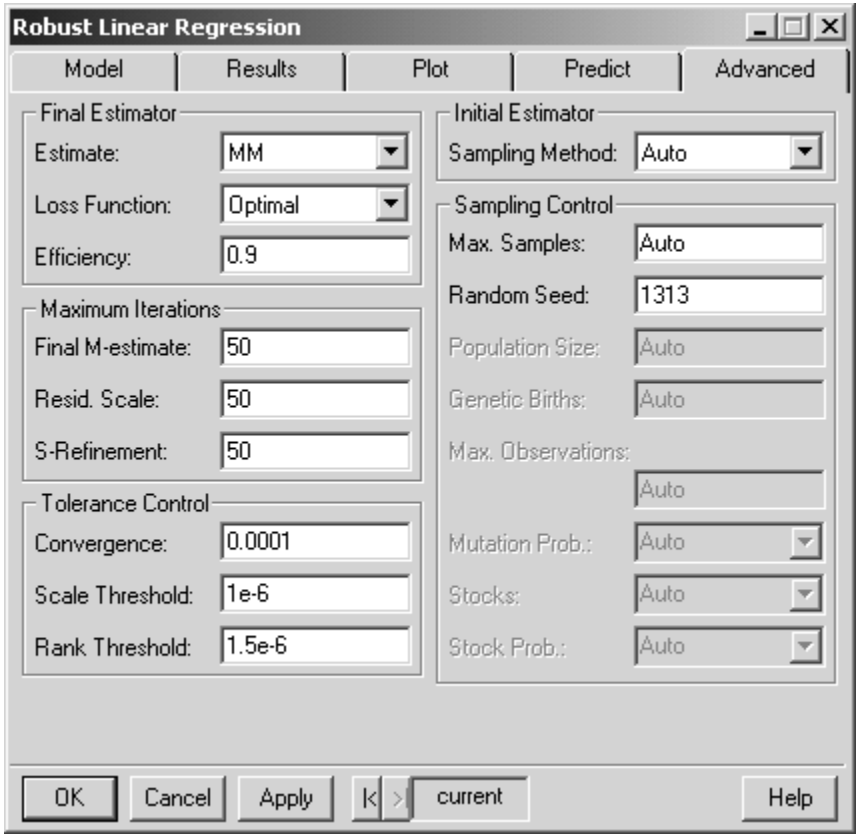


Figure 2.16: *The Robust Regression Advanced Page*

Adaptive Robust Estimate

For the final estimate, Spotfire S+ uses the MM-estimate briefly described in the section Overview of the Method: A Special M-Estimate on page 17. Alternatively, you can also use a very new *adaptive* robust estimate recently introduced by Gervini and Yohai (1999). This estimate has the property that in large samples it is fully efficient, i.e., has 100% efficiency, when the data is Gaussian. Thus the estimator performs equally as well as least squares in large samples. At the same time the estimator minimizes bias due to outliers nearly as well as the MM-estimate. See the section Theoretical Details for more information on this new estimator (not yet added in the current documentation).

To choose the adaptive robust estimate from the dialog, you simply click the down arrow to the right of the **Estimate** list box, and then select **Adaptive** from the drop-down list.

From the command line, you can use the `lmRob.robust.control` optional argument `final.alg`. For example,

```
> oil.tmp <- lmRob(Oil ~ Market, data = oilcity,
+ robust.control = lmRob.robust.control(final="adaptive"))
```

Efficiency at Gaussian Model

If the final MM-estimates are chosen, they have a default asymptotic efficiency of 90% compared with the LS estimates, when the errors are normally distributed. However, sometimes an efficiency of 90% may not be what you exactly want. For example, you might prefer 85% or 95%. Keep in mind that when you increase the efficiency from the default setting of 0.9 you get less protection from bias due to outliers, and conversely if you want more protection from bias due to outliers, use a smaller efficiency, e.g., 0.85 or 0.8.

To change the efficiency level, you can either type your desired Gaussian model efficiency in the **Efficiency** field, or use the `lmRob.robust.control` optional argument `efficiency` from the command line:

```
> oil.tmp <- lmRob(Oil ~ Market, data = oilcity,
+ robust.control = lmRob.robust.control(efficiency=0.95))
> coef(oil.tmp)

 (Intercept)    Market
  -0.0739893  0.8491134
```

M-Estimate Loss Function

The **Loss Function** list box in the **Final Estimator** region displays the default choice **Optimal**, indicating that Spotfire S+ uses as its default the *optimal* loss function ρ discovered by Yohai and Zamar (1998) for the final M-estimate (see Theoretical Details). This optimal loss function is shown in the upper right of Figure 2.17, and the corresponding psi-function $\psi = \rho'$ is shown in the lower right of that figure. The exact forms of the optimal ρ and ψ functions can be found in the section Theoretical Details.

If you wish, you can also choose to use the Tukey bisquare loss function ρ shown in the upper left of Figure 2.17 by selecting **Bisquare** from the drop-down list. The corresponding Tukey bisquare psi-function ψ is shown in the lower left of Figure 2.17.

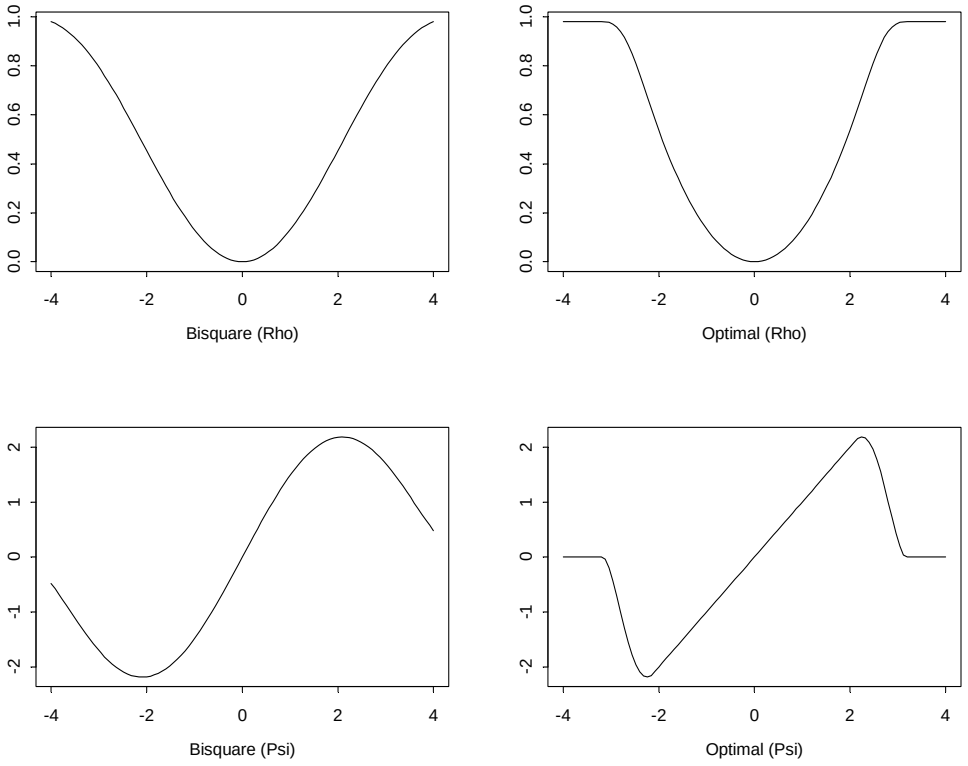


Figure 2.17: *Available Loss Functions*

To choose different settings of the loss function from the command line, you use the `lmRob.robust.control` optional argument `weight` as follows:

```
> control <- lmRob.robust.control(weight = c("Bisquare",
+ "Optimal"))
> oil.tmp <- lmRob(Oil ~ Market, data = oilcity,
+ robust.control = control)
```

```
> coef(oil.tmp)

 (Intercept)     Market
  -0.07914363  0.8450448
```

In the above commands, the rescaled bisquare function is used for the initial S-estimates, and the optimal loss function is used for the final M-estimates.

Optimizer Parameters

Describe maximum number of iterations and tolerance control constants here in the next release. The current default settings are reasonable.

Confidence Level of Bias Test

Spotfire S+ provides two bias tests for the default MM-estimate: one testing the bias of least squares coefficients against final M-estimates, and the other testing the bias of final M-estimate against the initial robust estimates.

To compute these tests for the model fit `oil.tmp` created in the above subsection, use the following command:

```
> test.lmRob(oil.tmp)

Test for Bias
            Statistics   P-value
 M-estimate       1.99 3.69e-001
LS-estimate      23.58 7.57e-006
```

The results show that the least squares estimate is biased relative to the final M-estimate, while the bias of the final M-estimates relative to the initial S-estimate is not significant.

By default, the level of significance of the tests is set at 10%. To change the level of the tests, you should specify the argument `level` for the `test.lmRob` function. A higher value of `level` will reject the final M-estimates more often, and a lower value of `level` will reject the final M-estimates less often.

Resampling Algorithms

When computing the initial S-estimates, Spotfire S+ uses an exhaustive resampling scheme for sufficiently small combinations of n and p, and otherwise uses a random resampling scheme when the number of numeric predictor variables p is no greater than 15. The random resampling scheme is designed so that a high breakdown point is achieved with high probability. When the number of numeric

predictors is greater than 15, S p o t f i r e S + uses a fast initial estimate due to Pena and Yohai (1999). You can over-ride this default if you wish, at both the GUI and the command line. You can also use a genetic algorithm for the initial S-estimate if you wish .

To choose a particular resampling algorithm from the dialog, you simply click the down arrow to the right of the **Sampling Method** list box, and then pick one from the drop-down list. These algorithms can also be selected from the command line by using the `initial.alg` argument to the function `lmRob.robust.control`, for which the valid choices are `"Random"`, `"Exhaustive"` and `"Genetic"`. Note that exhaustive resampling is only used/recommended when the sample size is small and there are less than 10 predictor variables.

Random Resampling Parameters

Random resampling is controlled by two parameters: a random seed and the number of subsamples to draw. By default, the number of subsamples is set at $[4.6 \cdot 2^p]$, where p is the number of explanatory variables, and $[\ ]$ denotes the operation of rounding a number to its closest integer. Note that this number will work fine if you have less than 15 predictor variables. However, if you have more than 15 predictor variables, the default number may be too big for computing in a reasonable time. To choose a different value for the number of subsamples to draw, use the optional argument `nrep` as follows:

```
> oil.tmp <- lmRob(Oil ~ Market,data = oilcity, nrep = 10)
```

The seed of the random resampling can be controlled by specifying the argument `seed` to `lmRob.robust.control`.

Genetic Algorithm Parameters

If you choose to use the genetic algorithm, the parameters for genetic algorithm can be changed through the `lmRob` optional argument `genetic.control`, the default of which is `NULL`. The optional argument `genetic.control` should be a list, usually returned by a call to the function `lmRob.genetic.control`. To look at the arguments of the function `lmRob.genetic.control`, use the following command:

```
> args(lmRob.genetic.control)
function(popsize = NULL, mutate.prob = NULL,
random.n = NULL, births.n = NULL, stock = list(),
maxslen = NULL, stockprob = NULL, nkeep = 1)
```

For an explanation of the various arguments above, see the online help file for the function `ltsreg.default`.

THEORETICAL DETAILS

Initial Estimate When p is Not Too Large

The key to obtaining a good local minimum of the M-estimation objective function when using a bounded, nonconvex loss function is to compute a highly robust initial estimate β^0. For models where the number of variables p is not too large, namely for p less than 15, Spotfire S+ does this by using the S-estimate method introduced by Rousseeuw and Yohai (1984). The S-estimate is part of an overall MM-estimate computational strategy proposed by Yohai, Stahel and Zamar (1991), and supported by a number of robustness experts who participated in the 1989 IMA summer conference on "Directions in Robust Statistics and Diagnostics."

The S-estimate approach has as its foundation an M-estimate $\hat{s}$ of an unknown scale parameter for observations $y_1, y_2, \ldots, y_n$, assumed to be robustly centered (that is, by subtracting a robust location estimate). The M-estimate $\hat{s}$ is obtained by solving the equation

$$\frac{1}{n}\sum_{i=1}^{n}\rho\left(\frac{y_i}{\hat{s}}\right) = .5 \tag{2.1}$$

where ρ is a symmetric, bounded function. It is known that such a scale estimate has a breakdown point of one-half (Huber, 1981), and that one can find min-max bias robust M-estimates of scale (Martin and Zamar, 1989, 1993).

The following regression S-estimate method was introduced by Rousseeuw and Yohai (1984). Consider the linear regression model modification of Equation (2.1):

$$\frac{1}{n-p}\sum_{i=1}^{n}\rho\left(\frac{y_i - x_i^T\beta}{\hat{s}(\beta)}\right) = .5 \tag{2.2}$$

For each value of β, we have a corresponding robust scale estimate $\hat{s}(\beta)$. The regression S-estimate (which stands for "minimizing a robust scale estimate") is the value $\hat{\beta}^0$ that minimizes $\hat{s}(\beta)$:

$$\hat{\beta}^0 = \text{argmin}_\beta \hat{s}(\beta) \tag{2.3}$$

This presents another nonlinear optimization, one for which the solution is traditionally found by a random resampling algorithm, followed by a local search, as described in Yohai, Stahel and Zamar (1991). Spotfire S+ allows you to use a genetic algorithm in place of the resampling algorithm, and also to use an exhaustive form of sampling algorithm for small problems. Once the initial S-estimate $\hat{\beta}^0$ is computed, the final M-estimate is obtained as the nearest local minimum of the M-estimate objective function.

For details on the numerical algorithms used, see Marazzi (1993), whose algorithms, routines and code were used in creating `lmRob`.

Fast Initial Estimate for Large p

When the number of variables p is 15 or greater, the above S-estimate based on random sample is often requires too much computation time for most users. Consequently Spotfire S+ uses a new "fast" initial estimate due to Pena and Yohai (1999). Although their new fast initial estimate is not guaranteed to have a high breakdown point, these authors provided evidence that the method favors well relative to other available robust estimates.

Alternating S and M Initial Estimate

For models with factor variables (with possibly many levels), Spotfire S+ uses a new initial estimate due to Maronna and Yohai. The new initial estimate uses an alternating resampling based S-estimate for the continuous (numeric) variables and a Huber type M-estimate with a least absolute deviations (LAD) start for the factor variables. This approach is based on the fact that for a model which contains only factor variables, there are no leverage points among the predictor variables and consequently LAD and Huber type M-estimates provide good initial parameter estimates. In this overall approach, the final estimate is the same as in the case of a linear model with only continuous (numeric) variables.

Optimal and Bisquare Rho and Psi-Functions

A robust M-estimate of regression coefficient β is obtained by minimizing

$$\sum_{i=1}^{n} \rho\left(\frac{y_i - x_i^T \beta}{\sigma}; c\right)$$

where $\rho(\cdot;c)$ is a convex weight function of the residuals with tuning constant c. The derivative of $\rho(\cdot;c)$ is usually denoted by $\psi(\cdot;c)$. For both the initial S-estimate and the final M-estimate in S-PLUS, two different weight functions can be used: Tukey's bisquare function and an optimal weight function introduced in Yohai and Zamar (1998).

Tukey's bisquare functions $\rho(\cdot;c)$ and $\psi(\cdot;c)$ are as follows:

$$\rho(r;c) = \begin{cases} \left(\frac{r}{c}\right)^6 - 3\left(\frac{r}{c}\right)^4 + 3\left(\frac{r}{c}\right)^2 & \text{if } |r| \le c \\ 1 & \text{if } |r| > c \end{cases}$$

$$\psi(r;c) = \begin{cases} \frac{6}{c}\left(\frac{r}{c}\right) - \frac{12}{c}\left(\frac{r}{c}\right)^3 + \frac{6}{c}\left(\frac{r}{c}\right)^5 & \text{if } |r| \le c \\ 1 & \text{if } |r| > c \end{cases}$$

The Yohai and Zamar optimal functions $\rho(\cdot;c)$ and $\psi(\cdot;c)$ are as follows:

$$\rho(r;c) = \begin{cases} 3.25c^2 & \text{if } \left|\frac{r}{c}\right| > 3 \\ c^2\left[1.792 + h_1\left(\frac{r}{c}\right)^2 + h_2\left(\frac{r}{c}\right)^4 + h_3\left(\frac{r}{c}\right)^6 + h_4\left(\frac{r}{c}\right)^8\right] & \text{if } 2 < \left|\frac{r}{c}\right| \le 3 \\ \frac{r^2}{2} & \text{if } \left|\frac{r}{c}\right| \le 2 \end{cases}$$

$$\psi(r;c) = \begin{cases} 0 & \text{if } \left|\frac{r}{c}\right| > 3 \\ c\left[g_1\frac{r}{c} + g_2\left(\frac{r}{c}\right)^3 + g_3\left(\frac{r}{c}\right)^5 + g_4\left(\frac{r}{c}\right)^7\right] & \text{if } 2 < \left|\frac{r}{c}\right| \le 3 \\ r & \text{if } \left|\frac{r}{c}\right| \le 2 \end{cases}$$

where

$$\begin{aligned} g_1 &= -1.944 & h_1 &= \frac{g_1}{2} \\ g_2 &= 1.728 & h_2 &= \frac{g_2}{4} \\ g_3 &= -0.312 & h_3 &= \frac{g_3}{6} \\ g_4 &= 0.016 & h_4 &= \frac{g_4}{8} \end{aligned}.$$

The Efficient Bias Robust Estimate

Yohai and Zamar (1998) showed that the ρ and ψ functions given above are optimal in the following highly desirable sense: the final M-estimate has a breakdown point of one-half, and minimizes the maximum bias under contamination distributions (locally for small fractions of contamination), subject to achieving a desired efficiency when the data are Gaussian.

Efficiency Control

The Gaussian efficiency of the final M-estimate is controlled by the choice of the tuning constant c. As discussed in the earlier sections, you can specify a desired Gaussian efficiency and Spotfire S+ automatically uses the correct c for achieving that efficiency.

Robust R-Squared

The robust R^2 is calculated as follows:

- *Initial S-estimator* $\hat{\beta}^0$

  If an intercept term is included in the model, then

  $$R^2 = \frac{(n-1)s_y^2 - (n-p)s_e^2}{(n-1)s_y^2}$$

  where $s_e = \hat{s}^0$ and s_y is the minimized $\hat{s}(\mu)$, for a regression model with only an intercept term with parameter μ. If there is no intercept term, replace $(n-1)s_y^2$ in the above formula with $n\hat{s}(0)^2$.

- *Final M-estimator* $\hat{\beta}^1$

  If an intercept term μ is included in the model, then

  $$R^2 = \frac{\sum\rho\left(\frac{y_i - \hat{\mu}}{\hat{s}^0}\right) - \sum\rho\left(\frac{y_i - x_i^T\hat{\beta}}{\hat{s}^0}\right)}{\sum\rho\left(\frac{y_i - \hat{\mu}}{\hat{s}^0}\right)}$$

  where $\hat{\mu}$ is the location M-estimate corresponding to the local minimum of

  $$Q_y(\mu) = \sum\rho\left(\frac{y_i - \mu}{\hat{s}^0}\right)$$

  such that

$$Q_y(\hat{\mu}) \leq Q_y(\mu^*)$$

where μ^* is the sample median estimate. If there is no intercept, replace $\hat{\mu}$ with zero in the formula.

Robust Deviance

For an M-estimate, the deviance is defined as the optimal value of the objective function on the σ^2-scale; that is:

- *Initial S-estimator* $\hat{\beta}^0$

$$D = \hat{s}^2(\hat{\beta}^0) = (\hat{s}^0)^2$$

- *Final M-estimator* $\hat{\beta}^1$

$$D = 2 \cdot (\hat{s}^0)^2 \cdot \sum \rho\left(\frac{y_i - x_i^T \hat{\beta}^1}{\hat{s}^0}\right)$$

Robust F Test

The robust F-statistics is

$$F_{ROB} = \frac{2}{p-q} \frac{1}{n} \sum_1^n \left(\rho\left(\frac{y_i - \underset{\sim}{x}_{q,i}^T \hat{\underset{\sim}{\beta}}_q}{\hat{\sigma}_p}\right) - \rho\left(\frac{y_i - \underset{\sim}{x}_{p,i}^T \hat{\underset{\sim}{\beta}}_p}{\hat{\sigma}_p}\right)\right)$$

where the subscript p indicates the predictor variables, coefficients, and robust residuals scale for the "full" p parameter model, and the subscript q indicates similar quantities for the "smaller" q parameter model, i.e. $q < p$.

Robust Wald Test

See Chapter 7 of Hampel, Ronchetti, Rousseeuw, and Stahel (1986).

Robust FPE (RFPE)

Ronchetti (1985) proposed to generalize the Akaike Information Criterion (AIC) to robust model selection. The results therein mimic the maximum likelihood approach thereby relying on an unbounded $\rho(\cdot;c)$. But such an estimate has a zero breakdown point. Yohai (1997) proposed an RFPE criterion which is a natural generalization of

Akaike's Final Prediction Error (FPE) criterion, and which is not tied to the likelihood approach. RFPE is based on the use of a bounded function $\rho(\cdot;c)$, and which should therefore retain a high breakdown point.

This new RFPE is calculated as follows:

$$RFPE \approx \left[\sum_{i=1}^{n} \rho\left(\frac{y_i - x_i^T \hat{\beta}}{\hat{s}^0} \right) \right] + p \frac{\hat{A}}{\hat{B}}$$

with

$$\hat{A} = \frac{1}{n} \sum_{i=1}^{n} \psi^2\left(\frac{r_i}{\hat{s}^0} \right) \qquad \hat{B} = \frac{1}{n} \sum_{i=1}^{n} \psi'\left(\frac{r_i}{\hat{s}^0} \right)$$

where $\hat{\beta}$ is the final M-estimate of β, and $r_i = y_i - x_i^T \hat{\beta}$. Note that when $\rho(u) = u^2 / 2$, RFPE reduces to the Akaike's classical FPE.

ROBUST GENERALIZED LINEAR MODELS

3

OVERVIEW OF THE METHODS

The Robust Library enables you to robustly fit Generalized Linear Models (GLIM's) for response observations y_i, $i = 1, \ldots, n$, that follow one of the following two distributions:

The Binomial Distribution

$$P(y_i = j) = \binom{n_i}{j} \mu_i^j (1 - \mu_i)^{(n_i - j)} \qquad j = 0, \ldots, n_i$$

where $0 \leq \mu_i \leq 1$ and n_i is the number of binomial trials for observation y_i. When $n_i = 1$ the observations are called y_i *Bernoulli* trials. The expected value of y_i for the Binomial distribution is related to μ_i by:

$$E\left(\frac{y_i}{n_i}\right) = \mu_i$$

The Poisson Distribution:

$$P(y_i = j) = \exp(-\mu_i)\ \mu_i^j / j! \qquad j = 0, \ldots, \infty$$

where $\mu_i > 0$. The expected value of y_i for the Poisson distribution is:

$$E(y_i) = \mu_i$$

In both of these cases you have a vector

$$x_i^T = (x_{i1}, \ldots, x_{ip})$$

of p independent explanatory variables, and corresponding vector

$$\beta^T = (\beta_1, \beta_2, \ldots, \beta_p)$$

of unknown regression coefficients, from which you form the linear predictor

$$\eta = x_i^T \beta .$$

The linear predictor η and the expected value μ_i are related through the *link* function g which maps μ_i to η :

$$\eta = g(\mu_i) .$$

The inverse link transformation g^{-1} maps η to μ_i:

$$\mu_i = g^{-1}(\eta) .$$

The robust library currently allows you to use (only) the canonical links for the Binomial and Poisson Families:

Binomial Model Canonical Link (the Logit link):

$$\eta = g(\mu_i) = \log\left(\frac{\mu_i}{1-\mu_i}\right), \qquad 0 < \mu_i < 1$$

with inverse transformation

$$\mu_i = g^{-1}(\eta) = \frac{\exp(\eta)}{1+\exp(\eta)}, \qquad (-\infty < \eta < \infty) \quad .$$

Poisson Model Canonical Link:

$$\eta = g(\mu_i) = \log(\mu_i), \qquad 0 < \mu_i < \infty$$

with inverse transformation

$$\mu_i = g^{-1}(\eta) = \exp(\eta), \qquad -(\infty < \eta < \infty)$$

For the Binomial model you have conditional expectation

$$E_\beta(y_i|x_i) = n_i \cdot \mu_i = n_i \cdot \frac{\exp(x_i^T\beta)}{1+\exp(x_i^T\beta)}$$

and for the Poisson model you have conditional expectation

$$E_\beta(y_i|x_i) = \mu_i = \exp(x_i^T\beta)$$

The classical approach to fitting the above models is to compute a maximum likelihood estimate (MLE) $\hat{\beta}_{MLE}$ as a solution of the following estimating equation.:

$$\sum_{i=1}^{n} \frac{\partial \log P(y_i,\ x_i^T\beta)}{\partial\beta}\Bigg|_{\beta=\hat{\beta}_{MLE}} = 0\ .$$

These are nonlinear equations that are solved iteratively, as described in McCullagh and Nelder (1989).

The classical MLE's for generalized linear models can be highly influenced by outliers. In all of the above models the explanatory vectors x_i can be highly influential outliers, calling for the use of a robust alternative to the MLE. In the Bernoulli case, the response y_i is either 0 or 1, and so can not be an outlier. In the general Binomial model when n_i is large, the y_i can also be outliers in cases where the expected values of y_i / n are small, and in the Poisson model the y_i can take on arbitrarily large integer values and so can be outliers. Thus in the general Binomial and Poisson cases, influential y_i outliers also call for a robust alternative to the MLE.

You may use one of the following three robust model fitting methods, the first of which you can use for both Binomial and Poisson models, while you use the second and third only for the Bernoulli model.

Conditionally Unbiased Bounded Influence Estimate

You compute a *conditionally unbiased bounded influence* (CUBIF) estimate $\hat{\beta}$ as a solution of the following equation:

$$\sum_{i=1}^{n} \psi_{a_i}(y_i - c_i - g^{-1}(x_i'\hat{\theta})) = 0$$

for certain constants a_i and c_i that make the estimate $\hat{\beta}$ consistent. The inverse link function g^{-1} is given above for the Binomial and Poisson models. The function ψ_a is even, bounded and chosen to minimize the trace of the asymptotic covariance matrix of the estimator $\hat{\beta}$, subject to a bound on bias due to a small fraction of outliers. For more details on the choice of ψ_a and other elements in the equation above, see the Theoretical Details section below.

Mallows-Type Unbiased Bounded Influence Estimates

For the Bernoulli case you compute a Mallows-type unbiased bounded influence estimate (a Mallows estimate) as a solution $\hat{\beta}$: of the estimating equation

$$\sum_{i=1}^{n} w_i \cdot \quad x_i \cdot \quad (y_i - g^{-1}(x_i^T\hat{\beta})) = 0$$

where the weights $w_i = w(x_i^T\hat{C}^{-1}x_i)$, with $\hat{C}$ a robust covariance matrix estimate are decreasing functions of the robust Mahalanobis distance of the explanatory vectors x_i (robustly measured *leverages* of x_i). You can use one of two weight functions, the *Carroll* and the *Huber* type, with the *Carroll* type being the default. For further details see the Theoretical Details section below.

Consistent Mis-Classification Model Estimate

For the Bernoulli model, you may also compute a so-called *mis-classification model estimate* $\hat{\beta}$, as a solution of the estimating equation

$$\sum_{i=1}^{n} w_i^{mc} \cdot x_i \cdot (y_i - F(x_i^T \beta, \gamma)) = 0,$$

where F is given by the mis-classification model

$$P(y_i = 1 | x_i) = g^{-1}(x_i^T \beta) + \gamma \cdot [1 - 2g^{-1}(x_i^T \beta)] = F(x_i^T \beta, \gamma)$$

with g^{-1} the Binomial inverse link given above, and the weights w_i^{mc} are given by a formula provided in the Theoretical Details section. This estimator, introduced by Copas (1988), has properties similar to those of the Mallows-type unbiased bounded influence estimates.

COMPUTING MLE AND ROBUST FITS WITH THE NT/ WINDOWS GUI

Computing Both MLE and Robust Fits

You can easily compute both MLE and robust GLIM fits for the Binomial and Poisson models with canonical links from the NT/ Windows GUI. Let's do so for the data set `breslow.dat`, analyzed by Breslow (1996) using a Poisson GLIM to explain the number of epilepsy attacks patients have during a given time interval. Choose **Robust ► Generalized Linear Models** from the menubar, to open the dialog shown below. Type in `breslow.dat` for **Data Set**, and select `sumY` as the dependent variable, and type the formula `Age10 + Base4 * Trt` in the **Formula** field. Select the *Poisson* Family and the *log* Link in the Model region to the right. The variable `sumY`

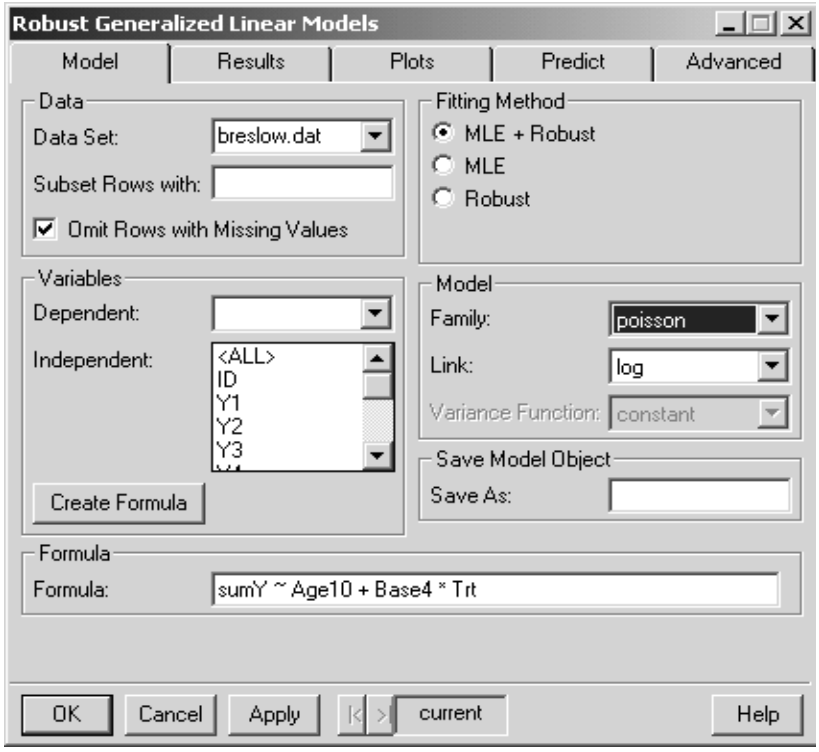


Figure 3.1: *The Robust Generalized Linear Models Dialog: Model Page*

contains the sum of the number of epilepsy attacks of each patient during the four weeks of the study, `Age10` contains the age of each patient divided by 10, `Base4` contains the base line number of epilepsy episodes in the four weeks prior to the study divided by 4, and `Trt` is a factor variable that indicates whether the patient received the active drug or the placebo.

The Model page is similar to the Model page for Generalized Linear Models except for the addition of the **Fitting Method** group. The default choice is **MLE + Robust** (both MLE and robust fits are computed) and the alternate choices are **MLE** (MLE only) and **Robust** (robust fit only). The **Advanced** page provides access to the various optional control features of the robust fit. The **Results** and **Predict** pages are identical to those of the Generalized Linear Models dialog. Also, you will notice some differences between the **Plots** page

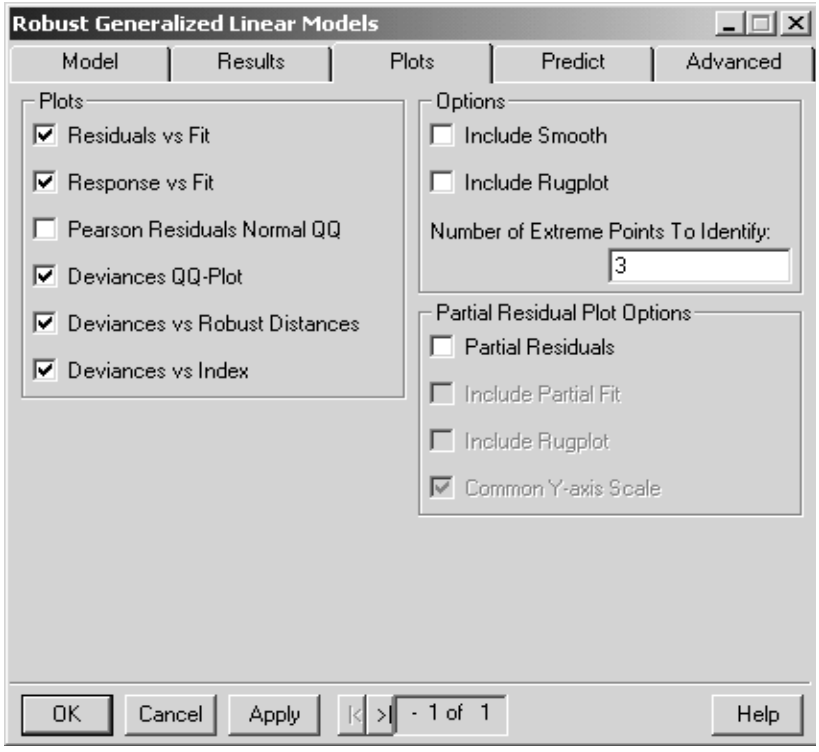


Figure 3.2: *The Robust Generalized Linear Models Dialog: Plots Page*

shown in Figure 3.2 and **Plots** page in the Generalized Linear Models dialog. Several new plots have been added, the Sqrt Abs Residuals vs Fit has been removed, and the Partial Residuals plots selection has been moved to the Partial Residuals Plot Options group.

The following three new plots are computed as defaults: the *Deviance QQ* plot, the *Deviances versus Robust Distances* plot, and the *Deviances versus Index* plot. The *Deviance QQ* plot is of particular note, having been added following a recent paper of Yohai and Garcia Ben (2000) that points out the inadequacy of normal QQ-plots of deviances. Since deviances are not typically well-approximated by a normal distribution, the deviance QQ-plot estimates the distribution of the deviances and plots the deviances against these estimated quantiles (see Yohai and Garcia Ben, 2000).

The *Deviances versus Robust Distances* plot gives information about the influence and outlyingness of the observations by plotting the deviances versus the robust distances of the corresponding predictor vectors, i.e., Mahalanobis distances based on a robust covariance matrix estimate for the predictor vectors. A large robust distance indicates that the observation has *leverage* and might overly influence the fit. See the corresponding section in Chapter 2 for more details.

In addition to the three plot types checked by default, check the *Residuals versus Fit* and *Response versus Fit* check-boxes. Click **OK** to compute both fits, along with the selected diagnostic plots. The results appear in a **Report** window and five tabbed pages of a **Graph Sheet**.

The Diagnostic Plots

Each of the **Graph Sheet** pages contains a Trellis display of the MLE and robust fits, as shown below.

Residuals versus Fitted Values

Figure 3.3 shows the Deviances vs. Fitted Values plots for both the MLE and robust fits. The MLE fit fails to identify the outlier, while the robust fit shows it clearly.

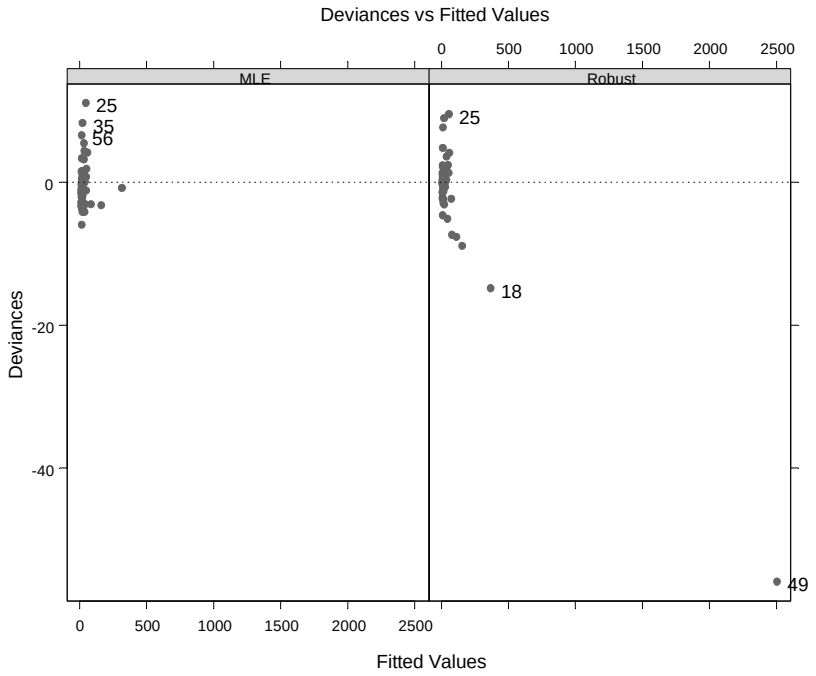


Figure 3.3: *Deviances vs Fitted Values*

Response versus Fitted Values

Figure 3.4 shows the Response vs. Fitted Values plots for both MLE and robust fits. Note that the MLE fitted line tries to accommodate the outlier, and consequently the corresponding deviance is small. On the other hand the robust fit downweights this observation, and achieves a fit that clearly reveals the anomaly.

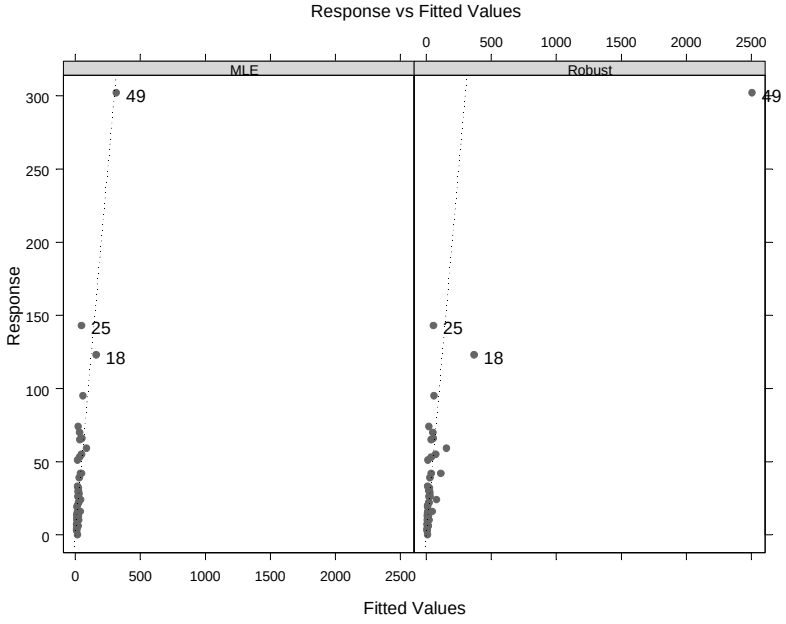


Figure 3.4: *Response vs Fitted Values*

Deviances QQ-Plot

In Figure 3.5 below, the deviances QQ-plot for the robust fit is shown side by side with the corresponding plot for the MLE fit.

We immediately identify an outlier in the robust deviances QQ-plot. In contrast, the MLE fit does not reveal any unusual observations. This illustrates one of the most important advantages of a robust fit relative to the MLE fit: the MLE fit is influenced by the outlier in such a way that the outlier is not revealed by the deviances.

.

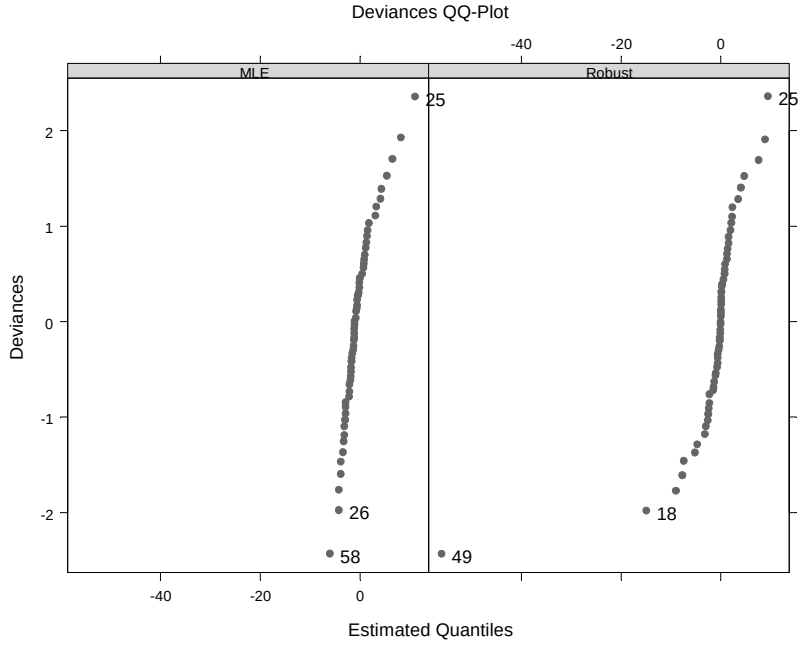


Figure 3.5: *Deviances QQ-Plot*

Deviances versus Robust Distances

Figure 3.6 shows a plot of the deviances versus the robust distances for both the MLE and robust fits. Both plots reveal that there are a number of predictor vectors that have high leverage. In the MLE plot only one of these, labeled 25, shows up as a large deviance, but in the robust plot, four of the high leverage points have large (negative) deviances.

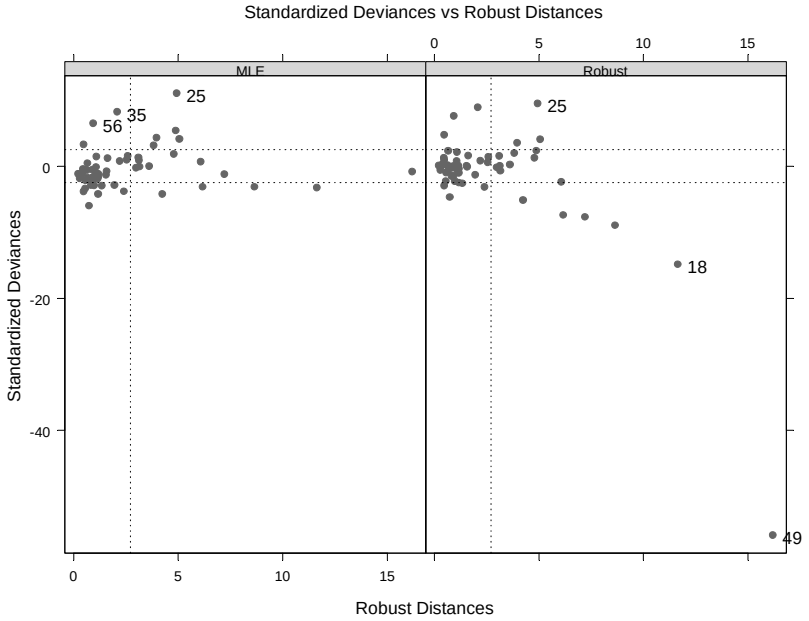


Figure 3.6: *Standardized Deviances vs Robust Distances*

Deviances versus Index

Figure 3.7 shows a plot of deviances versus observation index (row number) for both the MLE and robust fits. This plot lets you clearly see which observations are the outliers in the fits, and in particular those that are revealed by the robust fit but not by the MLE.

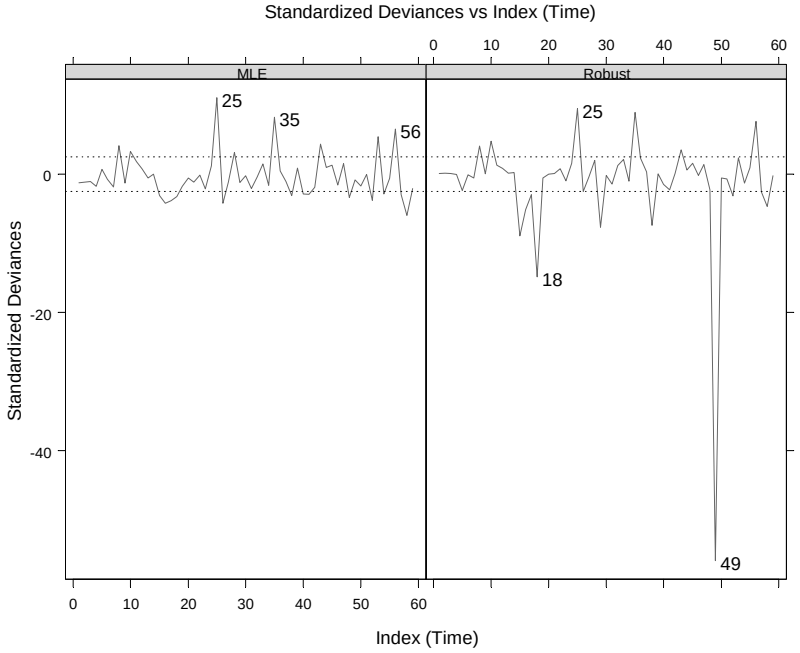


Figure 3.7: *Standardized Deviances vs Index (Time)*

The Statistics Report

The **Report** window contains the following results.

```
   *** GLM Fits Comparison ***

Calls:
Robust : glmRob(formula = sumY ~ Age10 + Base4 * Trt, family
= "poisson", data = breslow.dat, na.action = na.exclude)
   MLE : glm(formula = sumY ~ Age10 + Base4 * Trt, family =
"poisson", data = breslow.dat, na.action = na.exclude)
```

```
Residual Statistics:
              Min       1Q   Median       3Q      Max
Robust : -55.9440  -1.4577  -0.0307   1.0632   9.4756
   MLE :  -6.0032  -2.0744  -1.0803   0.8202  11.0386

Coefficients:
                         Value Std. Error  t value Pr(>|t|)
Robust : (Intercept)   1.6294   0.2709     6.0145   0.0000
   MLE : (Intercept)   1.8404   0.1298    14.1800   0.0000
Robust :       Age10   0.1286   0.0789     1.6305   0.1030
   MLE :       Age10   0.2435   0.0413     5.8967   0.0000
Robust :       Base4   0.1472   0.0224     6.5774   0.0000
   MLE :       Base4   0.0892   0.0022    40.4874   0.0000
Robust :         Trt  -0.2211   0.1169    -1.8914   0.0586
   MLE :         Trt  -0.1276   0.0383    -3.3359   0.0009
Robust :   Base4:Trt   0.0153   0.0221     0.6913   0.4894
   MLE :   Base4:Trt   0.0038   0.0022     1.7089   0.0875

Residual Deviance of model(s):
Robust : 3962
   MLE : 556.5
```

COMPUTING MLE AND ROBUST ESTIMATES AT THE COMMAND LINE

Computing Both MLE and Robust Fits

If you prefer to work at the command line, use the `fit.models` function in the **Robust Library** to compute both the MLE and the robust GLIM estimates and store them in a single S-PLUS object.

The different estimates (Conditionally Unbiased Bounded Influence, Mallows-type and Consistent Misclassification) are selected by specifying the argument `fit.method` of `glmRob`. You use `fit.method = 'cubif'`, `fit.method = 'mallows'` and `fit.method = 'misclass'` respectively. Recall that the options `mallows` and `misclass` only apply to the case of a Bernoulli response variable.

Conditionally Unbiased Bounded Influence

Compute both robust and MLE fits of a Poisson GLIM for the data set `breslow.dat`, and store the result as the fitted models object `breslow.fits` as follows:

```
> breslow.fits <- fit.models(list(Robust='glmRob',
+ MLE = 'glm'), sumY ~ Age10 + Base4 * Trt,
+ family = poisson, data = breslow.dat)
```

Now display a brief summary of the results

```
> breslow.fits

Calls:
Robust : glmRob(formula = sumY ~ Age10 + Base4 * Trt,
family = poisson, data = breslow.dat)
   MLE : glm(formula = sumY ~ Age10 + Base4 * Trt,
family = poisson, data = breslow.dat)

Coefficients:
             Robust     MLE
(Intercept)  1.6294  1.8404
      Age10  0.1286  0.2435
      Base4  0.1472  0.0892
        Trt -0.2211 -0.1276
  Base4:Trt  0.0153  0.0038
```

```
Residual Deviance Estimates:
Robust : 3962 on 54 degrees of freedom
   MLE : 556.5 on 54 degrees of freedom
```

Use the `summary` function to obtain more detailed information on the fit.

```
> summary(breslow.fits)

Calls:
Robust : glmRob(formula = sumY ~ Age10 + Base4 * Trt,
  family = poisson, data = breslow.dat)
   MLE : glm(formula = sumY ~ Age10 + Base4 * Trt,
  family = poisson, data = breslow.dat)

Residual Statistics:
              Min       1Q   Median       3Q      Max
Robust : -55.9440  -1.4577  -0.0307   1.0632   9.4756
   MLE :  -6.0032  -2.0744  -1.0803   0.8202  11.0386

Coefficients:
                       Value Std. Error  t value Pr(>|t|)
Robust : (Intercept)   1.6294   0.2709     6.0145   0.0000
   MLE : (Intercept)   1.8404   0.1298    14.1800   0.0000
Robust :       Age10   0.1286   0.0789     1.6305   0.1030
   MLE :       Age10   0.2435   0.0413     5.8967   0.0000
Robust :       Base4   0.1472   0.0224     6.5774   0.0000
   MLE :       Base4   0.0892   0.0022    40.4874   0.0000
Robust :         Trt  -0.2211   0.1169    -1.8914   0.0586
   MLE :         Trt  -0.1276   0.0383    -3.3359   0.0009
Robust :   Base4:Trt   0.0153   0.0221     0.6913   0.4894
   MLE :   Base4:Trt   0.0038   0.0022     1.7089   0.0875

Residual Deviance of model(s):
Robust : 3962
   MLE : 556.5
```

```
Correlations:
Robust:
            (Intercept)   Age10   Base4     Trt Base4:Trt
(Intercept)  1.0000
      Age10 -0.9022      1.0000
      Base4 -0.5218      0.1536  1.0000
        Trt -0.0472     -0.0254  0.3243  1.0000
  Base4:Trt  0.1536     -0.0115 -0.5477 -0.8985  1.0000

   MLE:
            (Intercept)   Age10   Base4     Trt Base4:Trt
(Intercept)  1.0000
      Age10 -0.9556      1.0000
      Base4 -0.4421      0.2234  1.0000
        Trt -0.0166     -0.0150  0.2401  1.0000
  Base4:Trt -0.1442      0.2260 -0.3834 -0.7784  1.0000The
Diagnostic Plots
```

Use the `plot` function to obtain diagnostic plots. When the command line menu appears as below, type 2 and 8 after "Selection:".

```
> plot(breslow.fits)

Make plot selections (or 0 to exit):

1: plot: All
2: plot: Deviances vs Fitted Values
3: plot: Response vs Fitted Values
4: plot: QQ-Plot of Pearson Residuals
5: plot: Deviances QQ-Plot
6: plot: Standardized Deviances vs Robust Distances
7: plot: Standardized Deviances vs Index (Time)
8: plot: Sqrt of abs(Deviances) vs Fitted Values
Selection(s): 3, 5
```

This results in the "Response vs Fitted Values" plots and the "Deviances QQ-Plot" in Figure 3.4 and Figure 3.5, respectively

You can also obtain any one of the plots in the menu with the `plot.glmRob` command, and appropriate arguments. For example, you get the Deviances QQ-plots with the command `plot.glmRob(my.glm.obj, which = 5)`.

Mallows-type estimate

Consider the data `leuk.dat` (Cook and Weisberg, 1982, p. 193). The data consist of measurements on 33 leukemia patients. The response variable is 1 if the patient survived more than 52 weeks. There are two covariates: WBC: White blood cell count, and AG: Presence or absence of certain morphologic characteristic in the white cells. You want to fit a Binomial GLIM.

Compute the MLE fit and a Mallows type fit with the default Carroll type weight function:

```
> leuk.mallows.default.fits <-fit.models(list(
+ Robust ='glmRob', MLE='glm'), y ~ wbc + ag,
+ family = binomial, data = leuk.dat,
+ fit.method = "mallows")
```

Now look at the brief summary of your fits,

```
> leuk.mallows.default.fits

Calls:
Robust : glmRob(formula = y ~ wbc + ag, family = binomial,
  data = leuk.dat, fit.method = "mallows")
   MLE : glm(formula = y ~ wbc + ag, family = binomial,
  data = leuk.dat)

Coefficients:
             Robust     MLE
(Intercept)  0.1710 -1.3074
        wbc -0.0002  0.0000
         ag  2.5240  2.2610

Residual Deviance Estimates:
Robust : 18.47 on 24 degrees of freedom
   MLE : 31.06 on 30 degrees of freedom
```

The Diagnostic Plots

Note that the coefficients of the robust Mallows type fit are different from those of the MLE. Cook and Weisberg (1982) detected that observation #15 was very influential in the model and that removing it yielded a better fit for the rest of the data. So make a deviance QQ-plots with:

```
> plot(leuk.mallows.default.fits, which = 4)
```

The deviance QQ-plot for the MLE in Figure 3.8 does not give much hint of an outlier, and is furthermore rather ragged and irregular. On the other hand, the deviance QQ-plot for the Mallows estimate in Figure 3.8 clearly reveals the outlier, and is otherwise more smoothly linear than in the case of the MLE. This large deviance in the case of the robust fit is due to the fact that the fitted probability for observation #15 is almost zero but the observed value is 1.

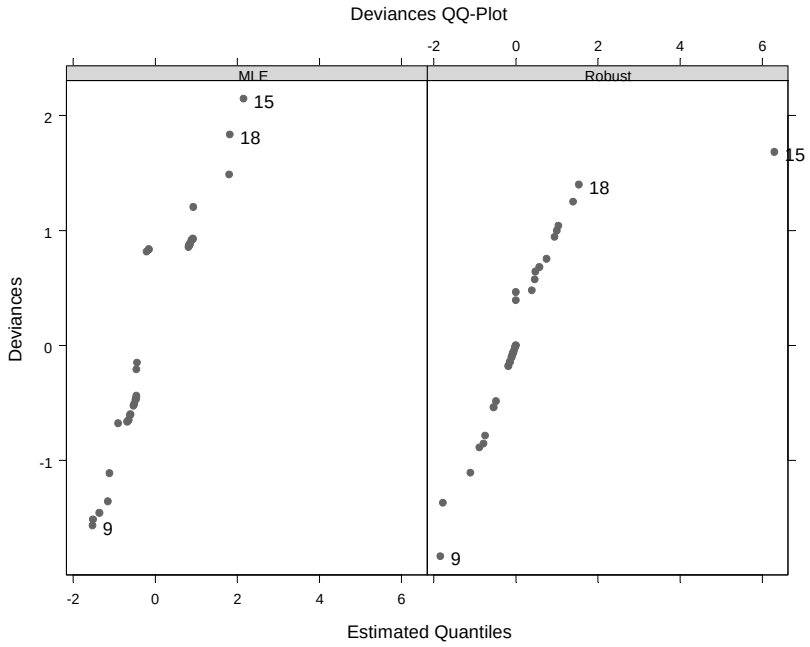


Figure 3.8: *Deviances QQ-Plot*

You can check to see if this observation automatically received weight zero with our Mallows-type estimate, by checking to see observations which received weight zero as follows:

```
> (1:33)[leuk.mallows.default.fits[[1]]$mallows.
+ weights == 0]
 [1] 15 16 17 31 32 33
```

Now you see that in fact observation #15 was removed from the fit.

To fit a Mallows-type estimate with Huber's weight function and tuning constant equal to 0.5, use

```
> leuk.mallows.huber.fits <- fit.models(list(
+ Robust = 'glmRob', MLE = 'glm'), y ~ wbc + ag,
+ family = binomial, data = leuk.dat,
+ fit.method = 'mallows', mallows.control =
+ glmRob.mallows.control(wt.fn = wt.huber, wt.tuning = .5))

> leuk.mallows.huber.fits

Calls:
Robust : glmRob(formula = y ~ wbc + ag, family = binomial,
  data = leuk.dat, fit.method = "mallows",
  mallows.control = glmRob.mallows.control(wt.fn =
  wt.huber, wt.tuning = 0.5))
   MLE : glm(formula = y ~ wbc + ag, family = binomial,
  data = leuk.dat)

Coefficients:
             Robust     MLE
(Intercept) -0.4913 -1.3074
        wbc -0.0001  0.0000
         ag  2.1547  2.2610

Residual Deviance Estimates:
Robust : 13.28 on 30 degrees of freedom
   MLE : 31.06 on 30 degrees of freedom
```

The shapes of the Carroll and Huber weight functions are different as indicated in Figure 3.9 and Figure 3.10, respectively. The former

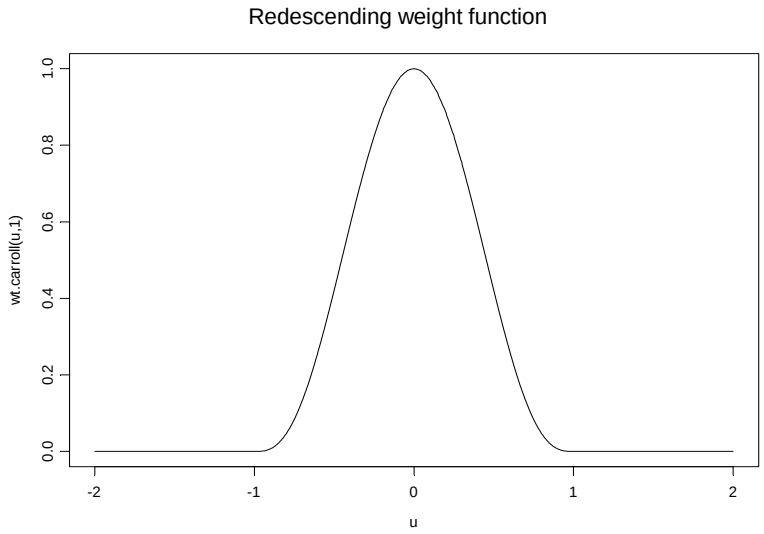


Figure 3.9: *Carroll's weight function*

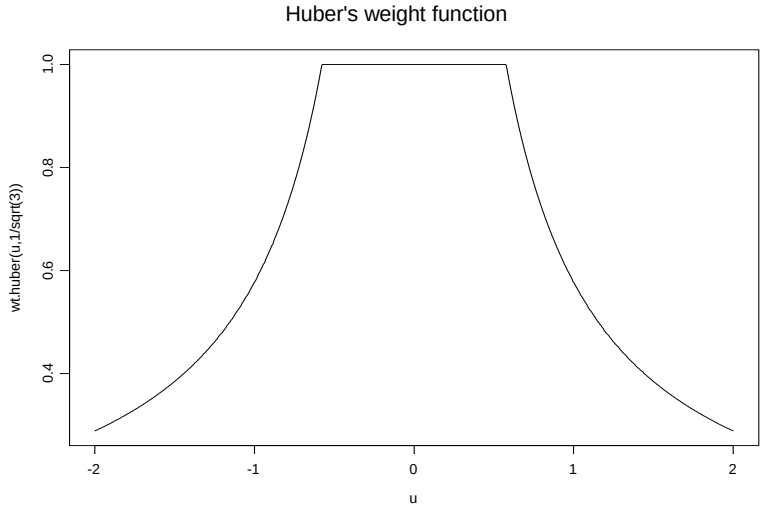


Figure 3.10: *Huber's weight function*

gives weight zero to sufficiently large observations, whereas the later gives weights that only approach zero for large argument values. This results in somewhat different values for the Mallows type estimates using these two weight functions. The Carroll type is preferred, which is why it is the default weight type. However, you may wish to experiment with both types.

Consistent Misclassification estimate

To fit a consistent misclassification estimate to the data analyzed above with misclassification probability $\gamma = 0.02$ use

```
> leuk.misclass.fits <- fit.models(list(Robust='glmRob',
+ MLE = 'glm'), y ~ wbc + ag, family = binomial,
+ data = leuk.dat, fit.method = "misclass",
+ misclass.control = glmRob.misclass.control(mc.gamma =
+ 0.02))

> leuk.misclass.fits

Calls:
Robust : glmRob(formula = y ~ wbc + ag, family = binomial,
  data = leuk.dat, fit.method = "misclass",
  misclass.control = glmRob.misclass.control(mc.gamma =
  0.02))
   MLE : glm(formula = y ~ wbc + ag, family = binomial,
  data = leuk.dat)

Coefficients:
             Robust     MLE
(Intercept)  0.1447 -1.3074
        wbc -0.0002  0.0000
         ag  2.4613  2.2610

Residual Deviance Estimates:
Robust : 17.66 on 30 degrees of freedom
   MLE : 31.06 on 30 degrees of freedom
```

Computing only a Robust GLIM Fit

Use the function `glmRob` to compute only the robust fit. For example, for the conditionally unbiased bounded influence example:

```
> breslow.robfit <- glmRob(sumY ~ Age10 + Base4*Trt,
+ family = poisson, data = breslow.dat)
```

```
> breslow.robfit
Call:
glmRob(formula = sumY ~ Age10 + Base4 * Trt, family =
  poisson, data = breslow.dat)

Coefficients:
 (Intercept)     Age10   Base4        Trt  Base4:Trt
    1.629352 0.1286164 0.14723 -0.2211265 0.01529229

Degrees of Freedom: 59 Total; 54 Residual
Residual Deviance: 3962.335
```

You can also use `plot` and `summary` to obtain more detailed information and plots, just as you would do with a "`glm`" object. The other estimates are fitted in a similar manner. For example, the Mallows-type estimate is obtained by typing

```
> leuk.mallows.default <- glmRob(y ~ wbc + ag,
+ family = binomial, data = leuk.dat,
+ fit.method = "mallows")
```

CONTROLLING OPTIONS FOR ROBUST GILM FITS

Conditionally Unbiased Bounded Influence

The user can set the parameters of the fitting algorithm for the Conditionally Unbiased Bounded Influence robust GLIM fits. These include: the maximum number of iterations, the tolerance of the convergence criterion, and the tuning constant of the final estimate. These control parameters can be set from the GUI by clicking the **Advanced** tab in Figure 3.1, or from the command line with the function `glmRob.control`. For example, in the commands shown below we set the maximum number of iterations to `50` and the precision of the convergence criteria to `10^-5`.

```
> breslow.control <- glmRob.cubif.control(maxit=50,
+ epsilon=1e-5)
> breslow.robfit <- glmRob(formula =
+ sumY ~ Age10 + Base4 * Trt, family = poisson,
+ data = breslow.dat, cubif.control = breslow.control)
> coef(breslow.robfit)

 (Intercept)     Age10    Base4        Trt Base4:Trt
    1.629497 0.1286316 0.147239 -0.2210817 0.0152693
```

The parameter `cpar` controls the initial estimate used in the iterative algorithm. By default it is set to 1.5. For more information see the section Theoretical Details below.

Mallows type estimate

For these estimates you can control the weighting function and its corresponding tuning constant. For example

```
> mallows.par <- glmRob.mallows.control(wt.fn = wt.huber,
+ wt.tuning = 3)
> mallows.rob <- glmRob(y~a + b + c, data = mallows.dat,
+ family = binomial, fit.method = 'mallows',
+ mallows.control = mallows.par)
```

The default values are `wt.fn = wt.carroll` and `wt.tuning = 8`. See section Theoretical Details below for more information on them.

Consistent Misclassification estimate

For these estimates you can set the probability of misclassification, the tolerance for the convergence criterion and the maximum number of iterations. You can also specify an initial value for the iterations and you can have `glmRob` print a trace of the current values of the parameters estimates while the algorithm iterates.

```
> misclass.par <- glmRob.misclass.control(mc.gamma = .01,
+ mc.maxit = 50, mc.tol = 1e-6, mc.trc = T)
> misclass.rob <- glmRob(y ~ ag + wbc, data = leuk.dat,
+ family = binomial, fit.method = 'misclass',
+ misclass.control = misclass.par)
```

The default values are `mc.gamma = 0.01`, `mc.maxit = 30`, `mc.trc = F`, `mc.tol = 0.001` and `mc.initial = NULL`.

THEORETICAL DETAILS

Conditionally Unbiased Bounded Influence Estimate

Consider general M-estimators defined implicitly by an estimating equation of the form

$$\sum_{i=1}^{n} \psi(y_i,\ x_i,\ \hat{\theta}) = 0.$$

In order to obtain consistent estimators we require that the above equation be unbiased. We say that an M-estimator is conditionally Fisher consistent if it satisfies

$$E_\theta(\psi(y,\ x,\ \theta)|x) = \int \int \psi(y,\ x,\ \theta) P_\theta(\partial y|x) = 0$$

for all θ and for all x. Pregibon (1981) and Stefanski, Carroll and Ruppert (1986) proposed M-estimates for Generalized Linear Models. However, those proposals are not conditionally Fisher consistent as defined above. Künsh, Stefanski and Carroll (1989) derived M-estimates that satisfy the above consistency condition and that are optimal in the following sense: they achieve minimum trace of the asymptotic covariance matrix subject to an upper bound on their *sensitivity*. Intuitively the *sensitivity* of an estimator measures the maximum influence that an arbitrary observation can have on any linear combination of the parameters (see Hampel *et al.* 1986). Künsh, Stefanski and Carroll (1989) also showed that the resulting estimator $\hat{\theta}$ is asymptotically normally distributed.

The estimate $\hat{\theta}$ is defined by the following system of equations

$$\sum_{i=1}^{n} \psi_{a_i}(y_i - c_i - g^{-1}(x_i'\hat{\theta}))x_i = 0 ,$$

$$\sum_{j=0}^{M_i} \psi_{a_i}(j - c_i - g^{-1}(x_i'\hat{\theta}))P(y_i = j|x_i) = 0 \qquad i = 1, \ldots, n ,$$

$$\frac{1}{n}\sum_{i=1}^{n} u_b(y_i, n_i, x_i'\hat{\theta}, c_i, |Ax_i|)(Ax_i)(Ax_i)' = I_p ,$$

where g^{-1} is the inverse of the link function; A is a p by p lower triangular matrix; $\psi_d(s) = \max[-d, \min(s, d)]$ is the Huber function with tuning constant d; $a_i = b/|Ax_i|$; b is a user chosen tuning constant that satisfies $b > \sqrt{p}$; I_p is the identity matrix of dimension p; $M_i = n_i$ in the Binomial case, and $M_i = \infty$ in the Poisson case. The function u_b is given by

$$u_b(y_i, n_i, x_i'\theta, c_i, |z_i|) = \sum_{j=0}^{M_i} [\psi_{b/|z_i|}(j - c_i - g^{-1}(x_i'\theta))]^2 P(y_i = j|x_i)$$

(see Marazzi, 1993). The fitting algorithm iterates updating estimates of c_i, A and θ. The initial estimates are Mallows estimates $\hat{\theta}_0$ and $\hat{A}_0$ that solve the following equations:

$$\sum \psi_c\left(\frac{\tilde{y}_i - x_i'\hat{\theta}_0}{\sigma}\right) w_i x_i = 0 ,$$

$$\sigma = med\left|\sqrt{w_i}(\tilde{y}_i - x_i'\hat{\theta}_0)\right| / \tau_0 ,$$

$$\frac{1}{n}\sum w_i^2 (\hat{A}_0 x_i)(\hat{A}_0 x_i)^t = I_p ,$$

$$w_i = \psi_b(\left|\hat{A}_0 x_i\right|) / \left|\hat{A}_0 x_i\right| ,$$

where τ_0 is chosen so that σ is consistent for normally distributed data, ψ_d is the Huber's function as above, and b and c are chosen by the user. The option `ufact` controls b in both sets of equations by setting $b =$ `ufact` $\times \sqrt{p}$ where p is the number of predictors. The constant c for the initial estimate is set with the option `cpar`.

Künsh, Stefanski and Carroll (1989) show that the estimate $\hat{\theta}$ is asymptotically normal. Its asymptotic covariance matrix can be consistently estimated by

$$(\hat{K} = \hat{S}_1^{-1}\ \hat{S}_2\ \hat{S}_1^{-1} / \ n) ,$$

where

$$\hat{S}_1 = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=0}^{M_i} \psi_{a_i}(j - c_i - n_i g(x_i'\hat{\theta}))P(y_i = j|x_i)(j - n_i g(x_i'\hat{\theta}))x_i x_i^t$$

$$\hat{S}_2 = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=0}^{M_i} \psi_{a_i}[(j - c_i - n_i g(x_i'\hat{\theta}))]^2 P(y_i = j|x_i)\ x_i x_i^t .$$

Mallows-type estimate

To calculate these estimates we need to asses the leverage of each observation x_i. Assume that $x_i = (1, z_i)$ where $z_i \in \Re^{p-1}$ and let $\eta_d(u)$ be a weight function that depends on a tuning constant d. Let (μ, V) be robust estimates of the center and dispersion matrix of the observations z_i (see the documentation for the `Splus` function `covRob` for details of these estimates). Let $d_i = (z_i - \mu_i)^t V^{-1}(z_i - \mu_i)$ be the Mahalanobis distances of the covariates to their centres. Define the weights w_i as

$$w_i = \eta_d(\sqrt{d_i / (p-1)}) .$$

Carroll and Pederson (1993) discuss the following choice for $\eta_d(u)$:

$$\eta_c(u) = \left(1 - \left(\frac{u}{c}\right)^2\right)^3 I(u \le c) ,$$

where

$$I(u \le c) = \begin{cases} 1 & \text{if } u \le c \\ 0 & \text{if } u > c \end{cases}$$

The choice $c = 8$ gives weight 0.75 or higher to those points with $d_i \le 2.5$, weight 0.50 or greater to those with $d_i \le 3.6$ and weight 0.25 or greater when $d_i \le 5$.

Another choice for the weights is based on Huber's function ψ. Let

$$\eta_c(u) = \begin{cases} 1 & \text{if } u \le c \\ u / c & \text{if } u > c \end{cases}$$

and

$$w_i = \eta_c(\sqrt{d_i / (p-1)}) ,$$

where c is $1/\sqrt{p-1}$ times the square root of the α-quantile of a χ^2_{p-1} distribution. Typically $\alpha = 0.95$ or $\alpha = 0.90$. To use these functions set `wt.fn = wt.huber` in `glmRob.mallows.control()`.

The default values in `glmRob.mallos.control()` are `wt.fn = wt.carroll` and `wt.tuning = 8`.

Carroll and Pederson (1993) show that the Mallows-type estimate is asymptotically normal and that its asymptotic covariance matrix can be consistently estimated by

$$\frac{1}{n} V_{n1}^{-1}(\hat{\theta}) V_{n2}(\hat{\theta}) V_{n1}^{-1}(\hat{\theta}) ,$$

where

$$V_{nj}(\theta) = \frac{1}{n} \sum_{i=1}^{n} w_i^j \, x_i \, x_i^t F^{(1)}(x_i^t \theta) , \qquad j = 1, 2$$

and $F^{(1)}(u)$ denotes the derivative of $F(u)$.

Consistent Misclassification Estimate

Copas (1988) proposed a misclassification model for the case of Bernoulli logistic regression. In this model each observation y_i is mistakenly classified with probability γ, $0 \leq \gamma \leq 1$, i.e.:

$$P(y_i = 1 \mid x_i) = F(x_i^t \theta) + \gamma [1 - 2F(x_i^t \theta)] = G(x_i^t \theta, \gamma) ,$$

where $F(u) = 1/(1 + \exp(-u))$ as before. The MLE estimator based on this model is not consistent to the parameters of the logistic model. Carroll and Pederson (1993) proposed the following simple procedure to obtain a consistent estimate based on the misclassification model.

Let $\hat{\theta}_{mc}$ satisfy

$$\sum_{i=1}^{n} w_i^{mc} \, x_i \, [y_i - G(x_i^t \hat{\theta}_{mc}, \gamma)] = 0 ,$$

with

$$w_i^{mc} = \frac{(1-2\gamma)F(x_i^t\hat{\theta}_{mc})(1-F(x_i^t\hat{\theta}_{mc}))}{G(x_i^t\hat{\theta}_{mc},\ \gamma)(1-G(x_i^t\hat{\theta}_{mc},\ \gamma))} .$$

Now, the corrected estimator $\hat{\theta}$ is defined by the following equation:

$$\sum_{i=1}^{n} w_i^{mc}\, x_i[y_i - F(x_i^t\hat{\theta})] = 0 .$$

To specify a particular value of γ use the option `mc.gamma` in `glmRob.misclass.control()`.

These estimates are determined by the choice of γ, the probability of misclassification. Carroll and Pederson (1993) show that these estimates have the same asymptotic behavior as the Mallows-type ones discussed above. They only difference is that in the above formulas for the asymptotic covariance matrix we have to use

$$w_i = w_i^{mc} = \frac{(1-2\gamma)F(x_i^t\hat{\theta})(1-F(x_i^t\hat{\theta}))}{G(x_i^t\hat{\theta},\ \gamma)(1-G(x_i^t\hat{\theta},\ \gamma))} .$$

ROBUST ANALYSIS OF VARIANCE

4

INTRODUCTION

This chapter shows you how to analyze designed experiments using robust model fitting as a complement to least squares model fitting. For designed experiments, it is usually assumed that experimental errors are normally distributed. In this case, the classical Analysis of Variance (ANOVA) technique based on least squares is safe to use. However, in many experiments the data may contain outliers that exert considerable undesirable influence on the least squares fit, and subsequent misleading analysis in the ANOVA context.

Outliers in data from industrial or laboratory experiments might be due to recording errors, or they might be valid and highly informative data points. For example, when industrial design experiments are performed, the independent variable design points may sometimes be set at rather extreme values in order to see how the dependent variable responds to extreme conditions. In such cases the response may take on extreme values that appear as outliers. Such data points should not be ignored as they may convey very important information about the response variable.

No matter what the cause of outliers, it is highly desirable to have a good robust model fitting method that fits the majority of the data well. Then outliers will be clearly exposed in the residuals for further study, no matter what the cause of the outliers. The robust ANOVA method that you will learn to use in this chapter accomplishes exactly that purpose. We remark that the robust fitting method will often lead to a better model choice with regard to the inclusion or exclusion of interaction terms. The reason is that even a single outlier can result in apparent significant interaction effects when fitting by the classical least squares method, and this is not the case when using a robust fit.

The robust ANOVA model fitting is carried out using an MM-estimate as in the case of robust linear regression described in Chapter 2, Robust Linear Regression, with one important difference. Because there are no independent variable leverage points in designed experiments, the computationally expensive initial estimate based on resampling is not required. Instead an L1 initial estimate is used.

FITTING LS AND ROBUST ANOVA MODELS WITH THE NT/WINDOWS GUI

This section shows you how to fit a robust ANOVA model using the NT/Windows GUI dialog.

The Lawson Data Set

The data object `lawson.dat` in the Robust Library consists of sixteen measurements from an unreplicated 2^4 designed experiment. View this data object via the GUI as follows. Display the Robust Library data sets in the left-hand pane of the Object Browser by one of the methods recommended in the Introduction chapter. Then double click on the `lawson.dat` icon to display the data sheet shown below:

		1	2	3	4	5
		C1	C2	C3	C4	y
1		N	N	N	N	47.46
2		P	N	N	N	49.62
3		N	P	N	N	43.13
4		P	P	N	N	46.31
5		N	N	P	N	51.47
6		P	N	P	N	48.49
7		N	P	P	N	49.34
8		P	P	P	N	46.10
9		N	N	N	P	46.76
10		P	N	N	P	48.56
11		N	P	N	P	44.83
12		P	P	N	P	44.45
13		N	N	P	P	59.15
14		P	N	P	P	51.33
15		N	P	P	P	47.02
16		P	P	P	P	47.90

Figure 4.1: *The Lawson Data Set*

You do not spot any unusual observations by simply looking at the data in tabular form.

Computing Both Least Squares and Robust Fits

You easily compute both classical and robust ANOVA fits to the `lawson.dat` data set using the fixed effects ANOVA dialog in the **Robust Library**.

Choose **Robust ► Fixed Effects ANOVA** from the menubar. The dialog shown below appears.

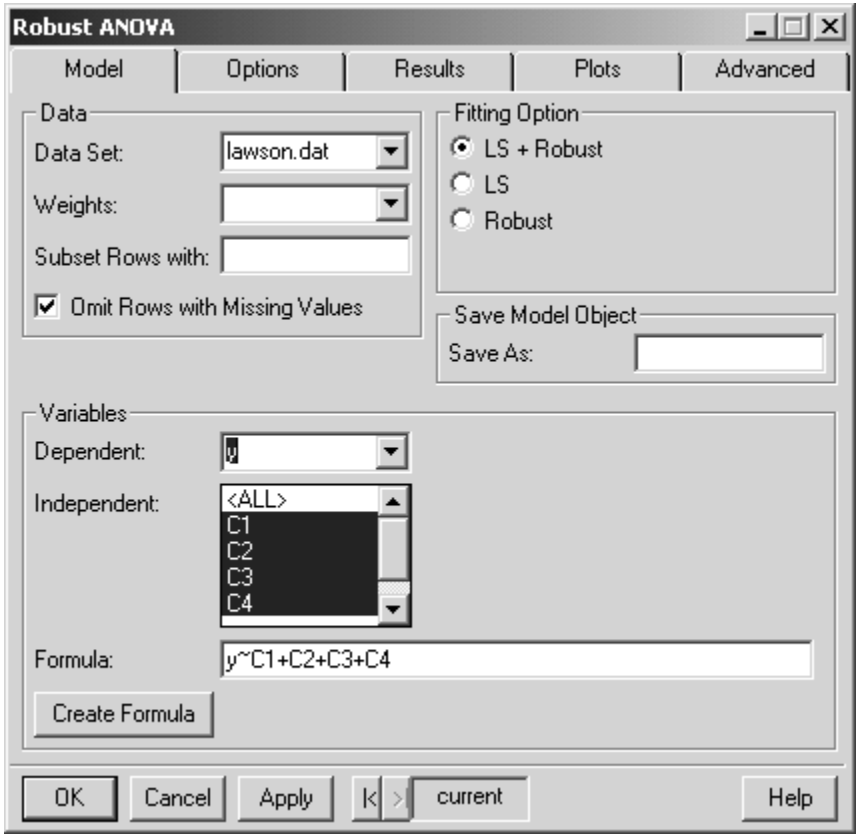


Figure 4.2: *The Robust ANOVA Dialog: Model Page*

Note that the **Robust ANOVA** dialog is identical to the (classical) **ANOVA** dialog under **Statistics ► ANOVA ► Fixed Effects**, except for the **Fitting Option** group at the upper right of the **Model** page,

some differences on the **Plot** page, and the **Advanced** tab which replaces the **Compare** tab. The default fitting option **LS + Robust** computes robust and least squares fits and ANOVA, **LS** computes only with a least squares fit, and **Robust** computes with a robust fit.

Type in `lawson.dat` under **Data Set** of the **Model** page, select `y` from the **Dependent Variable** drop-down list, and select ALL from the **Independent Variable** drop-down list. The formula `y~.` automatically appears in the **Formula** box, where y is the dependent variable and the "." on the right-hand side of the "~" indicates that all the four independent variables (factors) `C1`, `C2`, `C3`, and `C4` are included in the model. (Alternatively, select your dependent and independent variables in the Object Explorer before opening the **Robust ANOVA** dialog, as described in the Robust Linear Regression chapter).

Now click the **Plot** page tab to view the Plot page below.

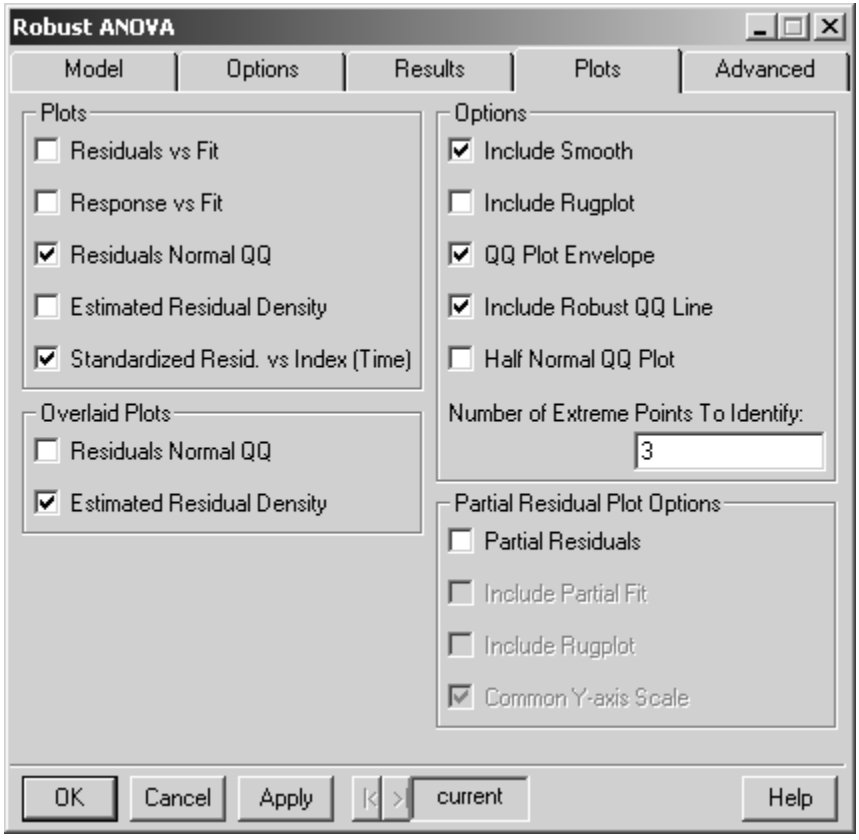


Figure 4.3: *The Robust ANOVA Dialog: Plots Page*

Note the checked boxes for the *Residuals Normal QQ*, *Estimated Residual Density* and *Standardized Resid. vs Index (Time)* plots. Uncheck the *Estimated Residual Density* box in the **Plots** region, and check the *Estimated Residuals Density* box in the **Overlaid Plots** region.

Click **OK**. This results in the computation of an ANOVA table for the Robust and LS fits, and three diagnostic plots corresponding to the above plot selections.

The Report Window

First look at the Report Window, which shows the following two ANOVA tables, one for the LS fit and one for the robust fit, with values arranged to facilitate easy comparison of the LS versus robust results:

```
Calls:
Robust : aovRob(formula = y ~ C1 + C2 + C3 + C4,
  data = lawson.dat, na.action = na.exclude)
    LS : aov(formula = y ~ C1 + C2 + C3 + C4,
  data = lawson.dat, na.action = na.exclude)

Comparison of ANOVA Tables:
          Df Sum of Sq Mean Sq RobustF   Pr(F)
Robust C1  1    0.3535  0.3535  0.1920  0.6552
    LS C1  1    2.5600  2.5600  0.3797  0.5503
Robust C2  1   39.0057 39.0057  5.5157  0.0167
    LS C2  1   71.2336 71.2336 10.5646  0.0077
Robust C3  1   27.3054 27.3054  8.5384  0.0029
    LS C3  1   55.0564 55.0564  8.1654  0.0156
Robust C4  1    0.0305  0.0305  0.0066  0.9341
    LS C4  1    4.0804  4.0804  0.6052  0.4530
```

Note that the robust F and p-values for the robust fit are rather different than those from the classical least squares fit. For details on the robust F and p-values, see the Theoretical Details section.

The Diagnostic Plots

From the above ANOVA tables, you see that the classical analysis of variance produces different results from the robust analysis, and you wonder whether this is caused by one or more outliers in the data. You quickly answer this question by looking at the three diagnostic plots you selected, which give you a quick visual comparison of results of the robust and classical fits. You find these plots on the three tabbed graph sheet pages, as shown below.

Look at normal QQ-plots for the LS and Robust residuals shown in Figure 4.4.

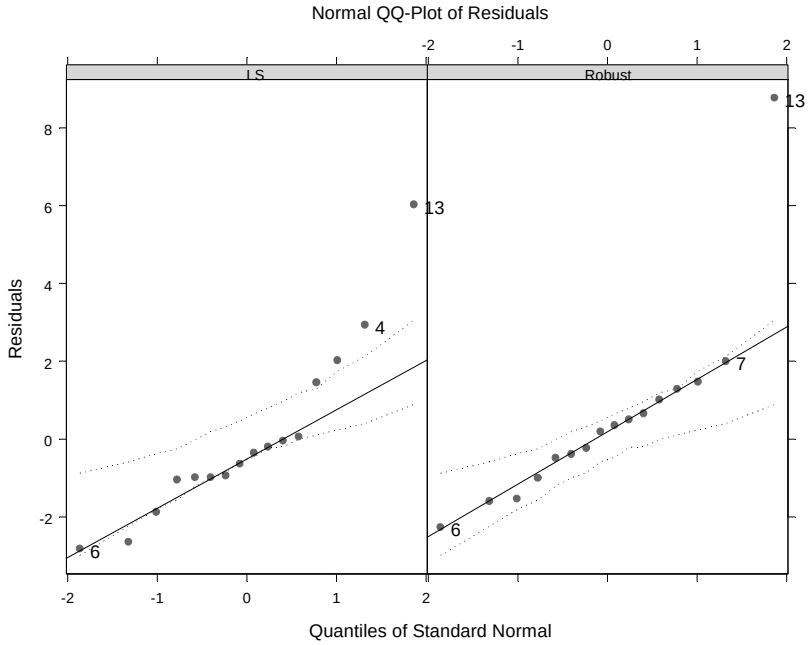


Figure 4.4: *Normal QQ-Plots for LS and Robust Residuals*

The dotted lines are 95% simulation envelopes. For the classical least squares fit you conclude that the residuals are approximately normally distributed, except for one moderately sized outlier and three marginal outliers in the right hand part of the left panel. However, the normal qq-plot of residuals from the robust fit in the right hand panel shows that there is really only one residual outlier, corresponding to the 13th observation, and that all the other residuals conform quite well to a normal distribution. For details on the 95% simulation envelopes, see the Theoretical Details section.

The plots of Standardized Residuals versus Index (in this case observation number) for the LS and Robust fits are shown in Figure 4.5 below. The horizontal reference lines at ±2.5 correspond to tail probabilities of .006 for a standard normal random variable. Note that the LS residuals barely hint at the presence of an outlier, while the

robust residuals clearly identify observation #13 as an outlier. These plots are consistent with the behavior of the normal QQ-plots in Figure 4.4.

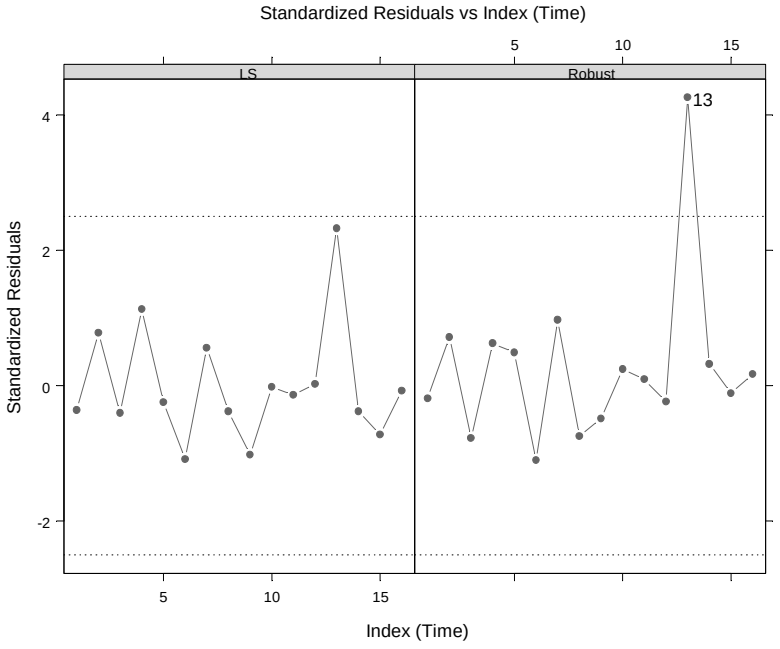


Figure 4.5: *Standardized Residuals vs Index for the LS and Robust fits*

Figure 4.6 below shows overlaid probability density estimates for the residuals from the least squares and robust fits. The density estimate for the latter gives a much more accurate picture of the distribution of the error term in the model: The main mode of the density estimate for the Robust residuals is well centered on zero, and has a single bump in the right-hand tail reflecting the presence of a single large positive outlier. On the other hand for the least squares fit, the density estimate main mode is shifted to the left of the origin, and there are two misleading bumps to the right.

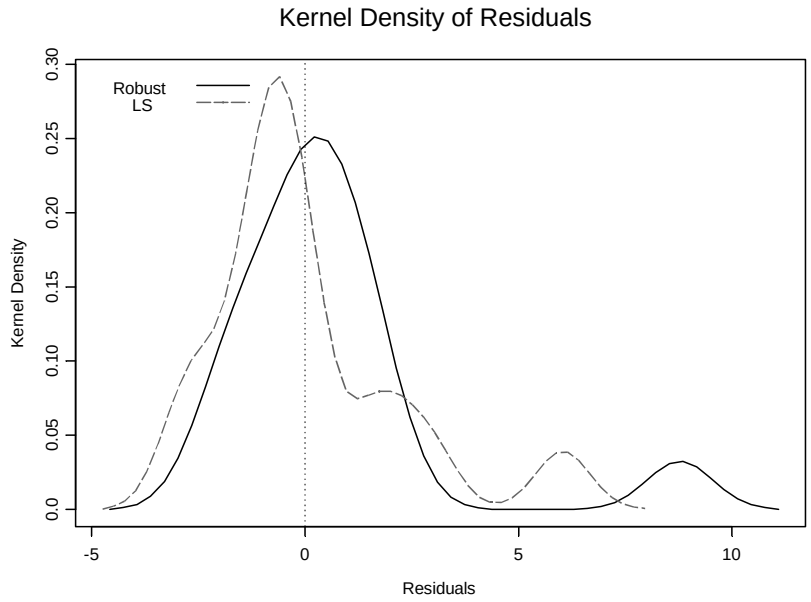


Figure 4.6: *Least Squares (Classical) and Robust Residuals Density Estimates*

COMPUTING ROBUST ANOVA AT THE COMMAND LINE

View the Data

View `lawson.dat` at the command line by typing `lawson.dat` and pressing **ENTER**:

```
> lawson.dat
   C1 C2 C3 C4     y
 1  N  N  N  N 47.46
 2  P  N  N  N 49.62
 3  N  P  N  N 43.13
 4  P  P  N  N 46.31
 5  N  N  P  N 51.47
 6  P  N  P  N 48.49
 7  N  P  P  N 49.34
 8  P  P  P  N 46.10
 9  N  N  N  P 46.76
10  P  N  N  P 48.56
11  N  P  N  P 44.83
12  P  P  N  P 44.45
13  N  N  P  P 59.15
14  P  N  P  P 51.33
15  N  P  P  P 47.02
16  P  P  P  P 47.90
```

Fitting LS and Robust ANOVA Models

Use the `fit.models` function in the **Robust Library** to fit two or more ANOVA models at once. For example, you fit ANOVA models for the `lawson.dat` data set with both least squares and robust fits as follows:

```
> lawson.both <- fit.models(list(Robust="aovRob",
+ Classical="aov"), y~., data=lawson.dat)
```

The function `aovRob` computes the robust fit and the function `aov` computes the LS fit. The returned object `lawson.both` is of class "`fit.models`". You use the `print`, `summary` and `plot` functions to compare different aspects of the two fitted models.

```
> lawson.both

Calls:
   Robust    aovRob(formula = y ~ ., data = lawson.dat)
Classical    aov(formula = y ~ ., data = lawson.dat)

Terms:
                              C1      ...       C4  Residuals
   Robust Sum of Squares  0.35348   ...   0.03047  98.24935
Classical Sum of Squares  2.56000   ...   4.08040  74.16920
   Robust Deg. ofFreedom        1      ...       1        11
Classical Deg. ofFreedom        1      ...       1        11

Residual Scale Estimates:
   Robust : 2.061 on 11 degrees of freedom
Classical : 2.597 on 11 degrees of freedom
```

Applying Generic Functions to Individual Models

Also, you can apply the print, summary and plot functions to individual fitted models in a "fit.models" object by using the models = optional argument, with the right-hand side a number indicating the position of the model in the "fit.models" object. For example, you view a summary of the robust fit as follows:

```
> summary(lawson.both, models = 1)

Calls:
Robust : aovRob(formula = y ~ ., data = lawson.dat)

Comparison of ANOVA Tables:
          Df Sum of Sq Mean Sq RobustF   Pr(F)
Robust C1  1    0.3535  0.3535  0.1920  0.6552
Robust C2  1   39.0057 39.0057  5.5157  0.0167
Robust C3  1   27.3054 27.3054  8.5384  0.0029
Robust C4  1    0.0305  0.0305  0.0066  0.9341
```

Here the `models =` optional argument specifies that you want a summary of the first fit only. Alternatively, you can call the print, summary and plot methods on the individual elements of the "fit.models" object. For example, you can make diagnostic plots for the robustly fitted model in `lawson.both` as follows:

```
> plot(lawson.both$Robust)

Make plot selections (or 0 to exit):

1: plot: All
2: plot: Normal QQ-Plot of Residuals
3: plot: Estimated Kernel Density of Residuals
4: plot: Residuals vs Fitted Values
5: plot: Sqrt of abs(Residuals) vs Fitted Values
6: plot: Response vs Fitted Values
7: plot: Residual-Fit Spread
8: plot: Standardized Residuals vs Index (Time)
9: plot: Overlaid Normal QQ-Plot of Residuals
10: plot: Overlaid Estimated Density of Residuals
Selection(s):
```

Fit Only a Robust ANOVA Model

The robust ANOVA method can also be invoked from the command line using the function `aovRob`. For example, you can use the following command to fit the same model as in the previous sections:

```
> lawson.rob <- aovRob(y~., data=lawson.dat)
```

The syntax for `aovRob` is the same as that for `aov`, and the returned object `lawson.rob` is of class "`aovRob`", which inherits from the class "`lmRob`". Typing the name of the object automatically invokes the print method giving a short summary of the fit:

```
> lawson.rob
Call:
aovRob(formula = y ~ ., data = lawson.dat)

Terms:
                C1       C2       C3       C4
 RobustF 0.191975 5.515712 8.538365 0.006575
Chisq Df        1        1        1        1

Robust residual scale: 2.060918
```

Use the summary function on this object to print the ANOVA table:

```
> summary(lawson.rob)
   Df   Sum of Sq     Mean Sq    RobustF       Pr(F)
C1  1  0.35348430  0.35348430 0.19197506 0.655245174
C2  1 39.00570248 39.00570248 5.51571219 0.016699961
C3  1 27.30537521 27.30537521 8.53836475 0.002904638
C4  1  0.03046612  0.03046612 0.00657517 0.934145352
```

The sums-of-squares are "robust sums-of-squares" and the F-test is a "robust F test". For information on these see the Theoretical Details section at the end of this chapter.

To view the diagnostic plots, use the command:

```
> plot(lawson.rob)

Make plot selections (or 0 to exit):

1: plot: All
2: plot: Normal QQ-Plot of Residuals
3: plot: Estimated Kernel Density of Residuals
4: plot: Residuals vs Fitted Values
5: plot: Sqrt of abs(Residuals) vs Fitted Values
6: plot: Response vs Fitted Values
7: plot: Residual-Fit Spread
8: plot: Standardized Residuals vs Index (Time)
9: plot: Overlaid Normal QQ-Plot of Residuals
10: plot: Overlaid Estimated Density of Residuals
Selection(s):
```

THEORETICAL DETAILS

Normal QQ-Plot Simulation Envelopes

The simulation envelopes are constructed by generating a 100-by-n matrix of independent standard normal samples. The rows of this matrix are then sorted so that the first column contains a sample of first order statistic, the second column contains a sample of the second order statistic, and so on. The third row and the 98^{th} row plotted against the sample quantiles form a roughly 95% quantile based confidence envelope for the observed values. For a more thorough discussion of simulation envelopes see Atkinson pp. 34-48.

Robust Sums-of Squares

The robust sums-of-squares are obtained by using the well-known expressions for sums-of-squares in terms of sums of squared model coefficients for balanced designs, using the robust model-coefficient estimates. For example in a two-way layout with I levels for main effects estimates α_i, J levels for main effects estimates $\hat{\beta}_j$, IJ interaction estimates γ_{ij}, and K observations per cell, the adjusted sums of squares is:

$$\sum_{i,\,j}(y_{ij.}-y_{..})^2 = J\cdot\sum_{i=1}^{I}\hat{\alpha}_i^2 + I\cdot\sum_{j=1}^{J}\hat{\beta}_j^2 + \sum_{i,\,j}\hat{\gamma}_{ij}^2$$

or

$$SS_{Y,\ adj} = SS_A + SS_B + SS_{AB}.$$

We compute robust sums-of-squares by using robust estimates in place of the usual least squares estimates of the main effects and interactions. We compute such robust sums-of-squares only for balanced designs.

Robust F-Tests

The robust F-test is of the same form as the robust F-test used for robust linear regression. See the Theoretical Details section of Chapter 2, Robust Linear Regression.

ROBUST COVARIANCE MATRIX ESTIMATION

5

OVERVIEW OF THE METHOD

The robust covariance estimators implemented in the **Robust Library** function `covRob` include the Fast MCD estimator, the Donoho-Stahel projection estimator, an M-estimator based on the Tukey biweight function that uses the Fast MCD for an initial estimate, and a new and relatively untested "pairwise" estimator (where the pairwise covariances are estimated using either the Gnanadesikan-Kettenring method or the quadrant correlation method) that adjusts the resulting covariance matrix to be positive definite. The default behavior is to choose an estimator from among the Donoho-Stahel, the Fast MCD and the quadrant correlation version of the pairwise estimtor based on the size of the data. If the data has dimension smaller than 1000 by 10 or smaller than 5000 by 5 the Donoho-Stahel is used. If the data has dimension greater than 50000 by 20 then the pairwise estimator is used. Otherwise (for medium sized problems), the Fast MCD is used.

We note that the robust correlation matrices are obtained from robust covariance matrices by dividing the latter by pairwise products of robust scale estimates obtained from the robust covariance matrix estimates.

Timing

The robust covariance estimators included in the Robust Library rely on computationally intense algorithms. The following tables compare the computation time required for each of the three estimators on several different sized data sets. These simulations were carried out on a Sun SPARC Ultra-60 with 1024MB of RAM.

Table 5.1: *MCD, Time in Seconds*

	n = 500	n = 1,000	n = 10,000	n = 50,000
p = 2	0.87	.77	1.24	3.96
p = 5	1.95	1.74	2.59	6.18
p = 10	4.18	3.52	5.40	10.98
p = 30	31.52	26.26	33.86	49.92

Table 5.2: *Donoho-Stahel estimator, Time in Seconds*

	n = 500	n = 1,000	n = 10,000	n = 50,000
p = 2	.84	1.87	31.04	
p = 5	.92	1.98	31.86	
p = 10	9.17	19.74	312.31	
p = 30				

Table 5.3: *M-estimator, Time in Seconds*

	n = 500	n = 1,000	n = 10,000	n = 50,000
p = 2	1.35	1.55	7.01	31.58
p = 5	3.28	2.75	12.01	61.26
p = 10	5.85	5.22	35.15	142.61
p = 30	35.79	34.80	129.96	496.00

NOTE: Timing results for the pairwise estimators will be provided here in a future release of the Robust Library. This estimator is quite new, and we encourage you to experiment with it and provide your feedback to us.

COMPUTING ROBUST COVARIANCE WITH THE WINDOWS GUI

Computing Both Classical and Robust Estimates

You can compute both classical and robust covariance or correlation estimates and compare the results using the Robust Covariance dialog. Try this out with the data set `woodmod.dat` which is included in the **Robust Library**. Display the **Robust Library** data sets in the left-hand pane of the Spotfire S+ Object Explorer using one of the methods recommended in the Introduction then select the data set `woodmod.dat`. Choose **Robust ▶ Covariance** (**Correlations**) **...** from the Spotfire S+ menubar to open the Robust Covariance dialog.

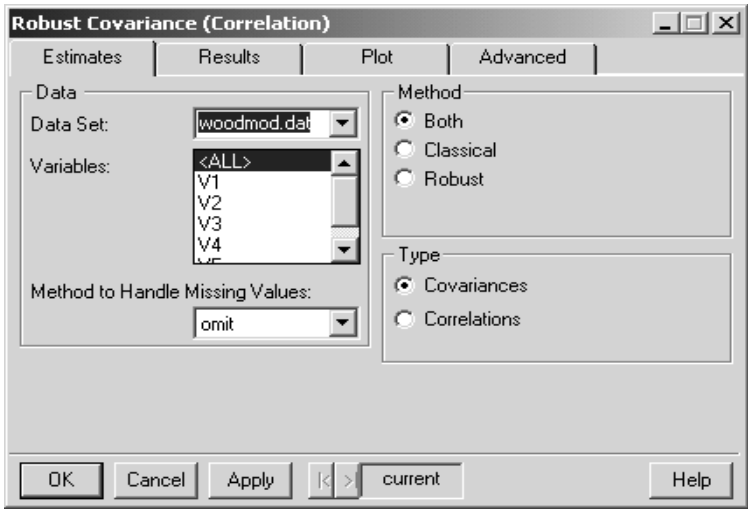


Figure 5.1: *Robust Covariance (Correlation) Dialog: Estimates Page*

NOTE: You could also skip selecting `woodmod.dat` in the Object Explorer, in which case you could type `woodmod.dat` into the **Data Set** combo box.

The default is to compute *both* classical and robust covariance estimates. You compute only one or the other by clicking the radio button of your choice in the **Method** region.

If you wish to compute covariances or correlations for only some of the variables in the data set, then select those variables in the **Variables** list.

To compute correlation estimates rather than covariance estimates, click the **Correlations** radio button in the **Type** region of the dialog.

Click on other page tabs to see what options are available for covariance estimation. For example, click the **Plot** page tab to reveal the plot options for the returned object:

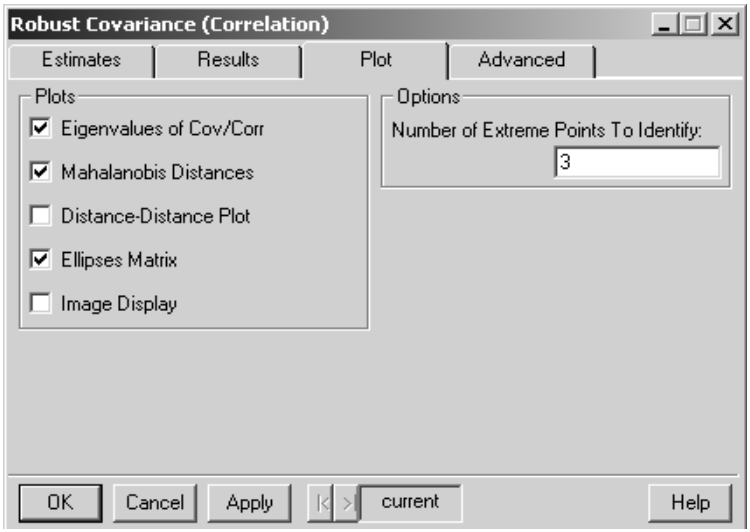


Figure 5.2: *Robust Covariance (Correlation) Dialog: Plot Page*

Eigenvalues Plot

The Eigenvalues comparison plot is shown in Figure 5.3. This plot is a convenient way for you to immediately see whether or not there is much difference between the classical and robust covariance (or correlation) estimates.

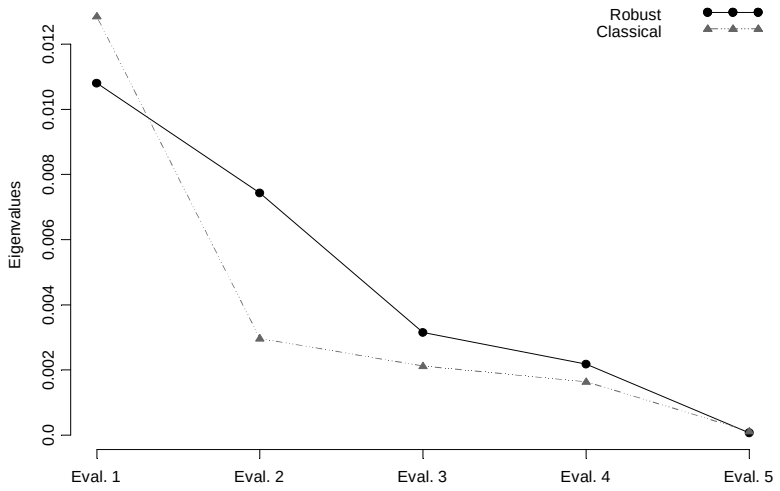


Figure 5.3: *Eigenvalues Plot*

Mahalanobis Distances

The Mahalanobis distances based on the classical and robust estimates are shown with Trellis display in Figure 5.4. From the distances based on classical covariance estimate, you do not see any outliers in the data set. In contrast, the robust distances using the default robust covariance estimate reveal quite a few outliers in the data set that are otherwise hard to find.

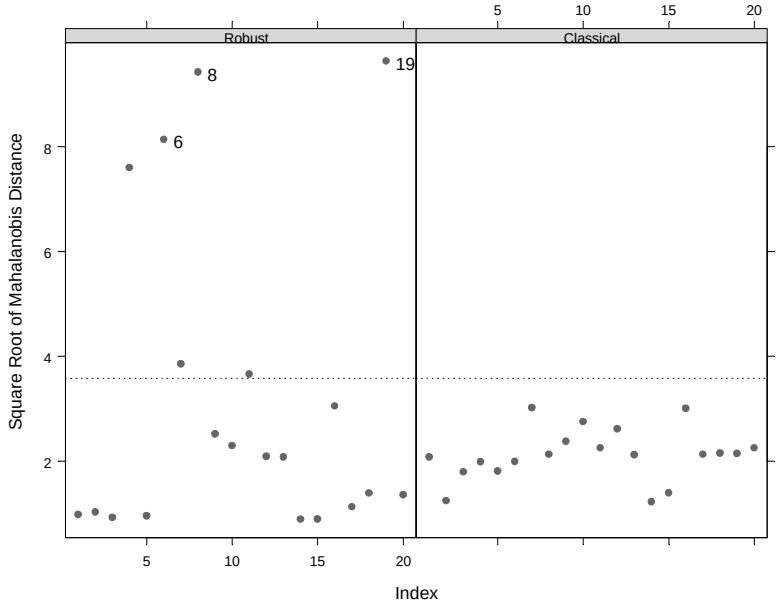


Figure 5.4: *Robust and Classical Mahalanobis Distances*

Distance-Distance Plot

The Distance-Distance Plot in Figure 5.5 plots the robust distances versus the classical Mahalanobis distances. The dashed line is the set of points where the robust distance is equal to the classical distance. The horizontal and vertical dotted lines are drawn at values equal to square root of the 97.5% quantile of a chi-squared distribution with p degrees of freedom. Points beyond these lines can be considered outliers.

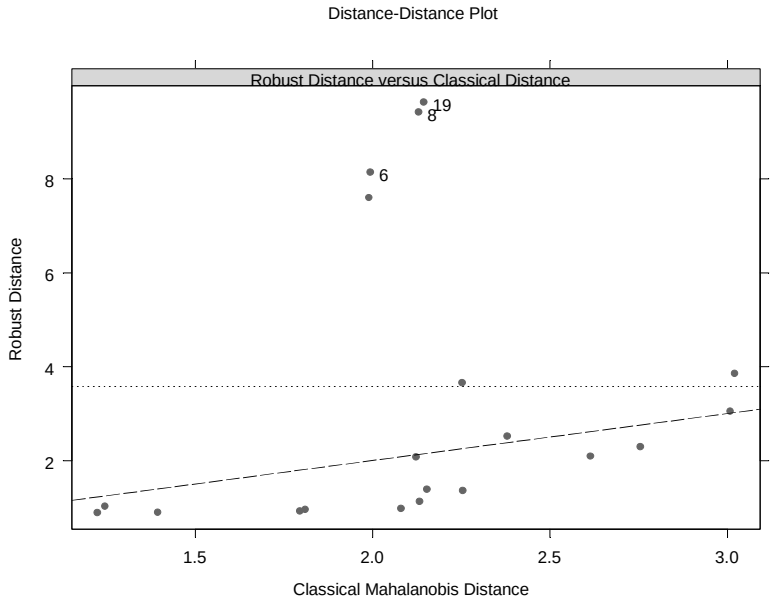


Figure 5.5: *Robust Distance vs. Classical Distance*

Visual Correlation Comparison Plot

This plot compares robust and classical correlation matrix estimates by interpreting the correlations in the upper triangle of each matrix as ellipses. The ellipses are drawn such that the i^{th}, j^{th} ellipse is the contour of a bivariate normal distribution with correlation ρ_{ij}. The lower triangle contains the numerical correlations. The overlaid ellipses are particularly useful for spotting where the robust and classical correlation estimates differ. In practice, this plot works best for matrices smaller than 10 by 10 and may still be useful for matrices as large as 25 by 25.

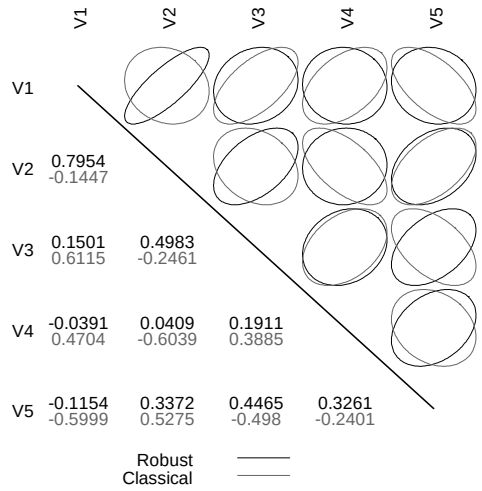


Figure 5.6: *Classical vs Robust Correlation Ellipses Plot*

Correlation Image Display

For sufficiently large covariance or correlation matrices the above display will no longer be useful because the ellipses and text will be too small. So we provide you with an image type display that gives an indication of pairs of variables for which the classical and robust correlations differ substantially. Specifically, the *image* function is used to display "significant" differences between robust and classical correlation matrix estimates as follows: the i-jth (j > i) element of the image matrix displays the standardized difference of the Fisher z-transformations of the robust and classical correlations for the two variables corresponding to i and j:

$$\text{image}_{i,\,j} = \frac{\left| Z_{i,\,j}^{\,rob} - Z_{i,\,j}^{\,cls} \right|}{\sqrt{2(n-p)}}$$

where n is the number of observations and p is the dimension of the data. Since Z^{rob} and Z^{cls} are approximately standard normal and typically positively correlated, 2(n-p) is conservative upper bound on the variance of their difference. The significance is calculated from

the quantiles of a standard normal so the values in the Correlation Image Display are almost certain to be more significant than the scale suggests.

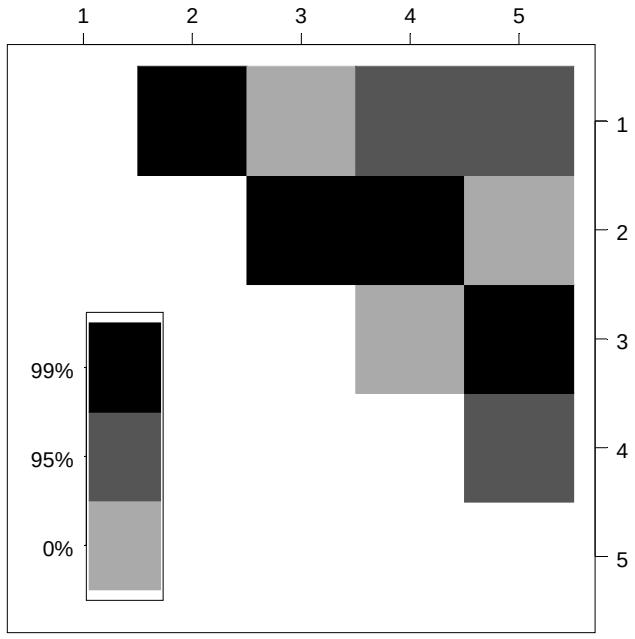


Figure 5.7: *Display of Significant Differences Between Classical and Robust Correlation Estimates*

Note that you interpret the legend intervals as follows: a cell value coded 0% has a value anywhere between zero and the 95th-percentile of the standard normal distribution; a cell value coded 95% has a value anywhere between the 95th and the 99th-percentile of the standard normal distribution; a cell value coded 99% has a value at least as large as the 99th-percentile of the standard normal distribution.

You can use the command line function *identify.cov* to view the robust and classical correlation estimates for a given cell. The function requires a graphsheet showing a Correlation Image Display and the `fit.models` object that created it. If the plot was created with the

GUI, the most recent object is saved as `.Last.guiCovRob` (unless you changed the default value in the *Save As* field). In this case, use the command,

```
> identify.cov(.Last.guiCovRob)
```

and select a cell from the display. For example, clicking in the *(2,3)* cell produces:

```
Correlation between V2 and V3:
        correlation
Robust:   0.7078387
    LS:  -0.2461404
```

NOTE: The above image display is a first attempt to display/compare large covariance matrix estimates. We hope to improve it in future releases of the Robust Library and we welcome any suggestions you may have.

The Statistics Report

The **Report** window output is shown below.

```
Calls:
   Robust : covRob(data = woodmod.dat, na.action = na.omit)
Classical : cov(data = woodmod.dat, na.action = na.omit)

Comparison of Covariance/Correlation Estimates:
 (unique correlation terms)
            [1,1]   [2,1]   [3,1]   [4,1]   [5,1]   [2,2]
   Robust  0.0102  0.0018  0.0011 -0.0002 -0.0008  0.0005
Classical  0.0083 -0.0003  0.0036  0.0027 -0.0029  0.0005

            [3,2]   [4,2]   [5,2]   [3,3]   [4,3]   [5,3]
   Robust  0.0008  0.0001  0.0005  0.0052  0.0008  0.0021
Classical -0.0004 -0.0008  0.0006  0.0042  0.0016 -0.0017

            [4,4]   [5,4]   [5,5]
   Robust  0.0035  0.0012  0.0042
Classical  0.0039 -0.0008  0.0028

Comparison of Location Estimates:
              V1     V2     V3     V4     V5
   Robust 0.5671 0.1165 0.5050 0.5520 0.9017
Classical 0.5509 0.1330 0.5087 0.5112 0.9070
```

```
Comparison of Eigenvalues:
          Eval. 1 Eval. 2 Eval. 3 Eval. 4 Eval. 5
   Robust 0.0108  0.0074  0.0031  0.0022  0.0001
Classical 0.0129  0.0030  0.0021  0.0016  0.0001
```

COMPUTING ROBUST COVARIANCE AT THE COMMAND LINE

Computing Both Classical and Robust Estimates

You can simultaneously compute classical and robust covariance (or correlation) estimates at the command line using the function `fit.models` in the **Robust Library**:

```
> woodmod.fm <- fit.models(list(Robust = "covRob",
+ Classical = "cov"), corr = T, data = woodmod.dat)

> woodmod.fm

Calls:
   Robust : covRob(data = woodmod.dat, corr = T)
Classical : cov(data = woodmod.dat, corr = T)

Comparison of Covariance/Correlation Estimates:
 (unique covariance terms)
            [1,1]   [2,1]   [3,1]   [4,1]   [5,1]   [2,2]
   Robust  0.0102  0.0018  0.0011 -0.0002 -0.0008  0.0005
Classical  1.0000 -0.1447  0.6115  0.4704 -0.5999  1.0000

            [3,2]   [4,2]   [5,2]   [3,3]   [4,3]   [5,3]
   Robust  0.0008  0.0001  0.0005  0.0052  0.0008  0.0021
Classical -0.2461 -0.6039  0.5275  1.0000  0.3885 -0.4980

            [4,4]   [5,4]   [5,5]
   Robust  0.0035  0.0012  0.0042
Classical  1.0000 -0.2401  1.0000

Comparison of Location Estimates:
              V1     V2     V3     V4     V5
   Robust 0.5671 0.1165 0.5050 0.5520 0.9017
Classical 0.5509 0.1330 0.5087 0.5112 0.9070
```

Note that the optional argument `corr=T` tells Spotfire S+ to compute the correlation matrix instead of the covariance matrix (the default, `corr=F`, is to compute a covariance estimate). The returned object is of class "`fit.models`". You can use the generic `plot` function on the returned object:

```
> plot(woodmod.fm)
```

which brings up the following menu of plot options for covariance models.

```
Make plot selections (or 0 to exit):

1: plot: All
2: plot: Eigenvalues of Covariance Estimate
3: plot: Sqrt of Mahalanobis Distances
4: plot: Ellipses Matrix
5: plot: Distance - Distance Plot
Selection(s):
```

The menu choices 2 through 5 produce the plots that appear in Figure 5.3, Figure 5.4, Figure 5.6, and Figure 5.5 respectively. `All` draws all four plots in order. You can create the Correlation Image Display described in the previous section using the function image.cov,

```
> image.cov(woodmod.fm)
```

and you can view the correlation of specific cells using identify.cov.

```
> identify.cov(woodmod.fm)
```

Finally, you can use the `summary` function on the "`fit.models`" object to produce the output shown in the Statistics Report above.

Computing Only One Estimate

Use the function `cov` if you just want a classical covariance or correlation matrix estimate, and use the function `covRob` if you just want a robust covariance matrix estimate. For example:

```
> covRob(stack.dat)
Call:
covRob(data = stack.dat)

Robust Estimate of Covariance:
               Loss Air.Flow Water.Temp Acid.Conc.
      Loss 30.35614 32.18611  12.727357  19.505018
  Air.Flow 32.18611 36.94189  12.223170  24.125233
Water.Temp 12.72736 12.22317   8.755008   9.727378
Acid.Conc. 19.50502 24.12523   9.727378  39.274340

Robust Estimate of Location:
     Loss Air.Flow Water.Temp Acid.Conc.
 13.83285 56.88652   20.51654    86.2826
```

CONTROLLING OPTIONS FOR ROBUST COVARIANCE ESTIMATION

In this section, you will learn how to change the default settings of the control parameters used by the robust covariance (correlation) estimators. You can change these options either through a GUI dialog (Windows only) or from the command line.

From the GUI Dialog

To change the default options for the robust covariance or correlation estimates from the GUI, click on the **Advanced tab** in the **Robust Covariance** (**Correlation**) dialog. This opens the **Advanced** page shown in Figure 5.8:

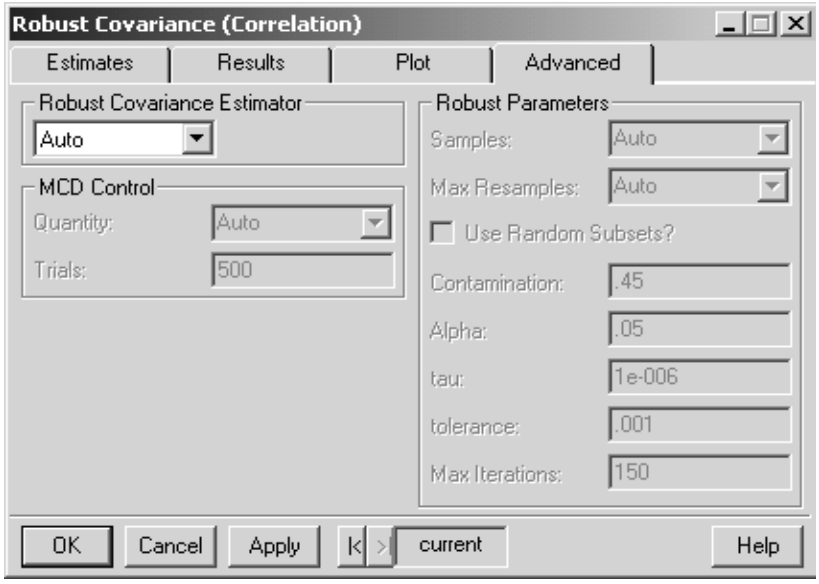


Figure 5.8: *Robust Control Parameters Dialog*

The control parameters are estimator specific, only the parameters relevant to the selected estimator are enabled in the dialog. Note that here are no control parameters for either of the pairwise estimators.

Table 5.4: *MCD Parameters*

GUI Option	Command Line Option	Description
Trials:	ntrial:	Number of subsets used for initial estimates. Default is 500.
Quantity:	quan:	The integer size of the subsets over which the determinant is minimized. Must be between the default is (n+p+1)/2 and the n. You may provide a fraction between .5 and 1, indicating the fraction of the data over which the determinant is minimized.

Table 5.5: *Donoho-Stahel Parameters*

GUI Option	Command Line Option	Description
Samples:	nresamp:	The number of resamples required (integer). If nresamp=0, all subsamples are taken. By default, nresamp is calculated so that P(high breakdown) = .99.
Max Resamples:	maxres:	The maximum number of resamples allowed.
Use Random Subsets?	random. sample:	A logical parameter. If TRUE, the current .Random.seed is used. If FALSE, the seed is fixed before the samples are drawn. For a specific seed x, at the command line use set.seed(x) and random.sample=T.
Tune:	tune:	The proportion of points assigned nonzero weight. Used to calculate the square root of the tune quantile of a chi squared distribution with p degrees of freedom.
Contamination:	eps:	Fraction of contamination used to calculate nresamp. By default, .5. If nresamp is given, eps is ignored.
Prob:	prob:	Probability of high breakdown used to calculate nresamp. Default is .99. Ignored if nresamp is given.

Table 5.6: *M-estimator Parameters*

GUI Option	Command Line Option	Description
Contamination:	r:	The fraction of contamination. The default is .45.
Alpha:	alpha:	Fraction of points receiving zero weight. By default, .05.
Max Iterations:	maxit:	The maximum number of M-iterations performed. By default, 150.
tolerance:	tol:	The relative precision of the solution of the M-estimate. The default is 1e-003.
tau:	tau:	The tolerance used to determine the singularity of the scatter matrix estimate. The default is 1e-006.

Remarks:

Since the M-estimator uses the MCD for an initial estimate, the MCD parameters are also relevant for the M-estimator.

The control option can also be used by any function that relies on covRob, namely princompRob and **discRob**.

Controlling Options at the Command Line

You can also change the default options for the robust covariance/correlation estimation from the command line. This involves using the covRob optional arguments estim, center, distance, and control as well as the function covRob.control.

estim

This argument allows you to specify the robust estimator used by covRob. The choices are "M" for a constrained M-estimate, "donostah" for a Donoho-Stahel estimate, "pairwiseQC" for a quadrant correlation based pairwise estimate, "pairwiseGK" for a Gnanadesikan-Kettenring based pairwise estimate, and "MCD" for the Fast MCD. The default value "Auto" chooses from among the Donoho-Stahel, the MCD and the quadrant correlation based pairwise estimator based on the size of the data.

`center`

If `center = T`, a robust location estimate is computed. If `center = F`, then the mean is assumed to be zero. A vector containing the mean may also be given. Note that `center` is only implemented for the Donoho-Stahel estimator.

`distance`

If `distance = T`, then the Mahalanobis distances are computed.

`control`

A list of control parameters for the estimator named in the covRob argument `estim`. The utility function `covRob.control` is useful for generating this list. It takes as arguments the name of the estimator (the same value assigned to `estim` in `covRob`) and the names and values of the control parameters you wish to specify. Any parameters not specified in the call to `covRob.control` will be assigned their default values. Refer to Figure 5.4, Figure 5.5, and Figure 5.6 for estimator specific parameters and their default values.

Examples:

Control parameters may be specified by using the function covRob.control,

```
> covRob(woodmod.dat, estim = "mcd",
+ control = covRob.control(estim = "mcd", ntrial = 250,
+ quan = 17))
```

or by passing them directly to covRob.

```
> covRob(woodmod.dat, estim = "mcd", ntrial = 250,
+ quan = 17))
```

The control argument is also useful for performing several analyses with the same list of parameters. Use `covRob.control` to make the list,

```
> rob.params <- covRob.control(estim = "donostah",
+ nresamp = 500)
```

then use rob.params as the control argument.

```
> covRob(woodmod.dat, estim = "donostah",
+ control = rob.params)
```

When using `fit.models` to fit multiple covariance/correlation models the control parameters must be specified through the `control` argument.

THEORETICAL DETAILS

MCD

The minimum covariance determinant estimator of location and covariance available in the Robust Library function `covRob` is similar to the existing S-PLUS function `cov.mcd`. This implementation uses the Fast MCD algorithm of Rousseeuw and Van Driessen (1999) to approximate the minimum covariance determinant estimator. This algorithm relies on a method called the "C-step" with which, given any approximation to the MCD, it is possible to compute another approximation with a smaller determinant.

Description of the C-step

Let $\boldsymbol{X} = (\boldsymbol{x}_1 \boldsymbol{x}_2 .. \boldsymbol{x}_n)^{\mathrm{T}}$ be a set of n data points in $\Re^p$. Suppose $H_j \subset \{1, \ldots, n\}$, $|H_j| = h$, and put $\boldsymbol{T}_j = \frac{1}{h} \sum_{i \in H_j} \boldsymbol{x}_i$ and $\boldsymbol{S}_j = \frac{1}{h} \sum_{i \in H_j} (\boldsymbol{x}_i - \boldsymbol{T}_1)(\boldsymbol{x}_i - \boldsymbol{T}_1)'$. If $\mathbf{det}(\boldsymbol{S}_j) \neq 0$, calculate the distances

$$d_j(i) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{T}_j)' \, \boldsymbol{S}_j^{-1} (\boldsymbol{x}_i - \boldsymbol{T}_j)}$$

and form a new subset H_{j+1} by choosing the h points with the smallest distances d_j. Then

$$\mathbf{det}(\boldsymbol{S}_{j+1}) \leq \mathbf{det}(\boldsymbol{S}_j)$$

with equality if and only if $\mathrm{T}_{j+1} = \mathrm{T}_j$ and $\mathrm{S}_{j+1} = \mathrm{S}_j$.

The Fast MCD algorithm

By default h is set equal to $[n+p+1]/2$ (h may be specified by using the control argument `quan`).

Repeat `ntrial` (by default, `ntrial = 500`) times:

- Draw a random $(p+1)$-subset J, and compute $\mathrm{T}_0 = \mathrm{mean}(\mathrm{J})$ and $\mathrm{S}_0 = \mathrm{cov}(\mathrm{J})$. If $\mathbf{det}(\boldsymbol{S}_0) = 0$ then extend J by adding another randomly chosen observation. Continue until $\mathbf{det}(\boldsymbol{S}_0) > 0$. Compute the relative distances (as in the C-step) and let H_1 be the set of h points with the h smallest distances.

- Carry out two C-steps and calculate det(S_3).

For the 10 results with the lowest det(S_3), continue taking C-steps until

$$\mathbf{det}(\boldsymbol{S}_{j+1}) = \mathbf{det}(\boldsymbol{S}_j)$$

then return the solution (T, S) with the minimum det(S).

Remark: When n is large, the initial 2 C-steps are done on a collection of up to 5 disjoint subsets containing at most 1500 data points.

For a complete description of the Fast MCD see Rousseeuw, P.J. and Van Driessen, K. (1999), A Fast Algorithm for the Minimum Covariance Determinant Estimator, *Technometrics*, 41, 212-223.

Donoho-Stahel

The Donoho-Stahel estimator is defined as a weighted mean and a weighted covariance matrix, where the weight of each point is a function of an "outlyingness" measure. The outlyingness measure r is based on the idea that if a point is a multivariate outlier then there must be some one-dimensional projection of the data for which the point is a univariate outlier. Suppose $\boldsymbol{X} = (\boldsymbol{x}_1 \boldsymbol{x}_2 .. \boldsymbol{x}_n)^{\mathrm{T}}$ is a set of n points in $\Re^p$. The outlyingness r of each point $\boldsymbol{x}_i$ is computed by finding the direction $\boldsymbol{a}_i \in \boldsymbol{A}$ where $\boldsymbol{A} = \{\boldsymbol{a} \in \Re^p \,|\, \|\boldsymbol{a}\| = 1\}$ such that

$$r_i = \sup_{\boldsymbol{a} \in \boldsymbol{A}} \frac{\left| \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{a} - \mathrm{med}\{\boldsymbol{x}_j^{\mathrm{T}} \boldsymbol{a}\}_{j=1}^{n} \right|}{\mathrm{mad}\{\boldsymbol{x}_j^{\mathrm{T}} \boldsymbol{a}\}_{j=1}^{n}}.$$

The weight w_i is computed using the following function of outlyingness:

$$w(r;c) = \begin{cases} 0 & \text{if } \left|\frac{r}{c}\right| > 1 \\ a_1 + a_2\left(\frac{r}{c}\right)^2 + a_3\left(\frac{r}{c}\right)^4 + a_4\left(\frac{r}{c}\right)^6 & \text{if } 0.8 < \left|\frac{r}{c}\right| \le 1 \\ 1 & \text{if } \left|\frac{r}{c}\right| \le 0.8 \end{cases}$$

where

$$a_1 = -19.71879$$
$$a_2 = 82.30453$$
$$a_3 = -105.45267$$
$$a_4 = 42.86694\,.$$

The tuning constant is set, by default, to be the square root of the .95 quantile of a chi squared distribution with p degrees of freedom.

The Donoho-Stahel estimator of location and scatter $(\boldsymbol{t}(\boldsymbol{X}), \boldsymbol{V}(\boldsymbol{X}))$ is defined as

$$\boldsymbol{t} = \boldsymbol{t}(\boldsymbol{X}) = \frac{\sum_{i=1}^{n} w_i \cdot \boldsymbol{x}_i}{\sum_{i=1}^{n} w_i}$$

and

$$\boldsymbol{V} = \boldsymbol{V}(\boldsymbol{X}) = \frac{\sum_{i=1}^{n} w_i (\boldsymbol{x}_i - \boldsymbol{t})(\boldsymbol{x}_i - \boldsymbol{t})'}{\sum_{i=1}^{n} w_i}$$

Computation

The practical difficulty with the Donoho-Stahel estimator lies in computing r. The algorithm implemented in the Robust Library uses an approximation based on subsampling. Define $\tilde{r}$ as r but where the supremum is taken over a finite set $\boldsymbol{A}$, defined as follows. For each subsample $\tilde{\boldsymbol{X}}$ of size p from $\boldsymbol{X}$, let $\boldsymbol{a}$ be the direction orthogonal to the hyperplane containing $\tilde{\boldsymbol{X}}$; let $\boldsymbol{A}$ be the set of all these $\boldsymbol{a}$'s. Since $\boldsymbol{A}$ will in general be too large to be useful for computation, one replaces $\boldsymbol{A}$ with a random subsample $\boldsymbol{A}_N$ of size N. The number of subsamples N is computed using the relation $1 - (1 - (1 - \varepsilon)^{p+1})^N = \texttt{prob}$ where ε (`eps`) is the desired breakdown

point and `prob` is the probability of not breaking down. By default we take $\varepsilon = 0.5$, `prob` $= 0.99$, and solve for N as a function of p. The weights are calculated using the resulting outlyingness measure $\tilde{r}_N$.

For the complete description see Maronna, R.A. and Yohai, V.J. (1995), The Behavior of the Stahel-Donoho Robust Multivariate Estimator, *Journal of the American Statistical Association*, Vol. 90, No. 429, 330-341.

M-estimator

Suppose $\boldsymbol{X} = (\boldsymbol{x}_1 \boldsymbol{x}_2 .. \boldsymbol{x}_n)^{\mathrm{T}}$ is a set of n points in R^p. An S-estimate of multivariate location and shape is defined as the vector $\hat{\boldsymbol{t}}$ and the positive definite symmetric matrix $\hat{\boldsymbol{C}}$ that minimize the determinant of $\hat{\boldsymbol{C}}$ subject to

$$(5.1) \qquad n^{-1} \sum_{i=1}^{n} \rho\left(\frac{[(\boldsymbol{x}_i - \boldsymbol{t})^{\mathrm{T}} \boldsymbol{C}^{-1} (\boldsymbol{x}_i - \boldsymbol{t})]^{1/2}}{c} \right) = n^{-1} \sum_{i=1}^{n} \rho\left(\frac{d_i}{c} \right) = b_0$$

where $d_i = [(\boldsymbol{x}_i - \boldsymbol{t})^{\mathrm{T}} \boldsymbol{C}^{-1} (\boldsymbol{x}_i - \boldsymbol{t})]^{1/2}$ and ρ is a non-decreasing function on $[0, \infty)$.

The function ρ is chosen to be a scaled version of a base function such as the biweight, which reaches a maximum of 1 at c_0. The constant b_0 is chosen as $b_0 = r\rho(c_0)$ for breakdown r which is set by default to 0.45. Since ρ reaches a maximum of 1 at c_0, b_0 equals 0.45 as well. The constant c is chosen so that the estimate $\hat{\boldsymbol{C}}$ of $\boldsymbol{C}$ is consistent under multivariate normality, that is such that $\mathrm{E}(\rho(d/c)) = b_0$ where the expectation is taken under a chi-squared distribution with p degrees of freedom.

Let $\tilde{d}_i = d_i / c$. An S-estimate is also a solution $(\hat{\boldsymbol{t}}, \hat{\boldsymbol{C}})$ of a weighted mean and covariance iteration

$$\hat{\boldsymbol{t}}_i^{(j)} = \frac{\sum w(\tilde{d}_i^{(j)})\boldsymbol{x}_i}{\sum w(\tilde{d}_i^{(j)})} \tag{5.2}$$

$$\hat{\boldsymbol{C}}^{(j)} = \frac{\sum w(\tilde{d}_i^{(j)})(\boldsymbol{x}_i - \boldsymbol{t}^j)(\boldsymbol{x}_i - \boldsymbol{t}^j)^{\mathrm{T}}}{\sum \upsilon(\tilde{d}_i^{(j)})} \tag{5.3}$$

where

$$w(\tilde{d}) = \psi(\tilde{d}) / \tilde{d}$$

$$\psi(\tilde{d}) = \rho'(\tilde{d})$$

$$\upsilon(\tilde{d}) = \tilde{d}\,\psi(\tilde{d}).$$

Note that zero weight is given when $\tilde{d}_i > c$.

The above iteration in turn can be viewed as an iterative computation of an M-estimate of location and covariance, starting with a highly robust initial estimate $(\hat{\boldsymbol{t}}^0, \hat{\boldsymbol{C}}^0)$. The M-estimator included in the Robust Library uses the Fast MCD as an initial robust estimate then refines the estimate with M iterations using the translated biweight function described below.

For the complete description of this estimator see Rocke, D.M. (1996), Robustness Properties of S-Estimators of Multivariate Location and Shape in high Dimension, *Annals of Statistics*, Vol 24, No. 3, 1327-1345.

Translated Biweight

Since zero weight is given to points with distance larger than c, one might expect that points that are a great distance from the main body of points will receive zero weight. This is the case for one-dimensional data where the 50% breakdown biweight S-estimator gives zero weight to any point $\boldsymbol{x}_i$ such that $\tilde{d}_i > 1.55$. However, this behavior changes as p increases. In 20 dimensions a point must lie at least a distance of $\sqrt{94.5}$ from the mean to receive zero weight from the 50% breakdown biweight S-estimator. Under normality, such distances occur with probability on the order of 10^{-11}. Points much closer to the center are clear outliers, but are still assigned positive weight in the analysis. Figure 5.9 shows the weight versus the distance $\tilde{d}$ for the biweight and the translated biweight. Also shown is the density of the

square root of a chi-squared variate with 10 degrees of freedom (the distribution of the $\tilde{d}$ for normal data). The biweight assigns positive weight to outliers and down-weights most of the "good" data.

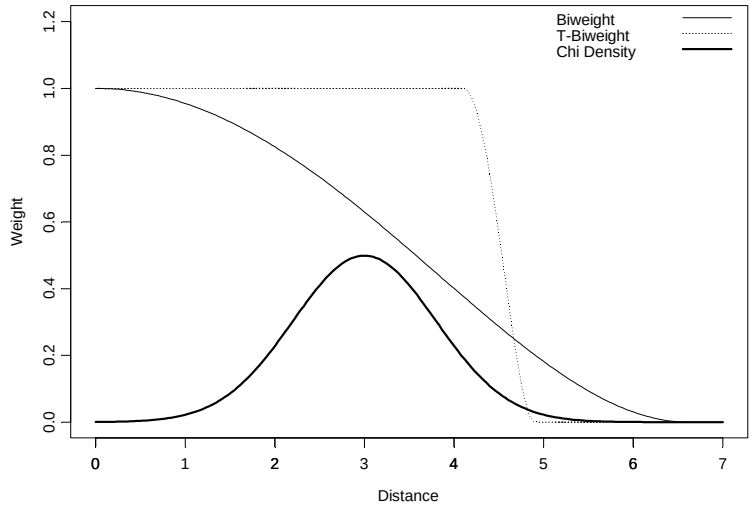


Figure 5.9: *The Biweight and Translated Biweight Functions*

This problem is addressed by using a weight function that is essentially the same as the biweight except that it has been translated. It is clear from figure 5.8 that the translated biweight assigns full weight to the major of the data and zero weight to any clear outliers. The translated biweight is defined by a two-parameter class of ρ functions.

$$(5.4) \quad w_t(d;\, c, M) = \begin{cases} 1 & 0 \le d < M \\ (1 - ((d - M)/\, c\,)^2)^2 & M \le d \le M + c \\ 0 & d > M + c \end{cases}$$

$$(5.5) \quad \psi_t(d;\, c, M) = \begin{cases} d & 0 \le d < M \\ d(1 - ((d - M)/\, c\,)^2)^2 & M \le d \le M + c \\ 0 & d > M + c \end{cases}$$

$$\rho_t(d;\, c, M) = \begin{cases} d^2/\ 2 & 0 \le d < M \\ f(d;c,\ M) & M \le d \le M + c \\ M/\ 2\ + c((5c + 16M)/\ 30\ ) & d > M + c \end{cases} \tag{5.6}$$

where

$$\begin{aligned} f(d;\, c, M) = {} & M^2/\ 2 \\ & - M^2(M^4 - 5M^2c^2 + 15c^4)/\ (30c^4) \\ & + d^2(1/\ 2 + M^4/\ (2c^4) - M^2/\ c^2) \\ & + d^3(4M/\ (3c^2) - 4M^3/\ (3c^4)) \\ & + \mathrm{d}^4(3M^2/\ (2c^4) - 1/\ (2c^2)) \\ & - 4Md^5/\ (5c^4) + d^6/(6c^4) \end{aligned} \tag{5.7}$$

The parameters c and M are chosen to give the desired breakdown point and asymptotic rejection probability (that is the probability that a "good" data point lies beyond the rejection point).

The Pairwise Robust Covariance Estimator

This estimator was recently proposed by Maronna and Zamar (2001) to allow you to compute robust covariance matrices "safely" with many more variables p than with the other estimators above. This estimator has complexity n in the sample size n, and p^2 in the number of variables p. The estimator computes all pairwise covariances using either the estimator proposed by Gnanadesikan and Kettenring (1972), or the quadrant correlation estimator, using a very clever adjustment to insure that the resulting covariance matrix is positive definite. Further details on the estimator will be provided in the next release of the Robust Library.

ROBUST PRINCIPAL COMPONENT ANALYSIS

6

COMPUTING ROBUST PRINCIPAL COMPONENTS WITH THE NT/WINDOWS GUI

Computing Both Classical and Robust Estimates

You can quickly compute both classical and robust principal components estimates and generate a comparison of the results using the Robust Principal Component Analysis dialog. Try this out with the data set `woodmod.dat` which is included with the **Robust Library**. First select the data set `woodmod.dat` in the Spotfire S+ Object Explorer. Then choose **Robust ▶ Principal Components ...** from the Spotfire S+ menubar to open the following dialog box.

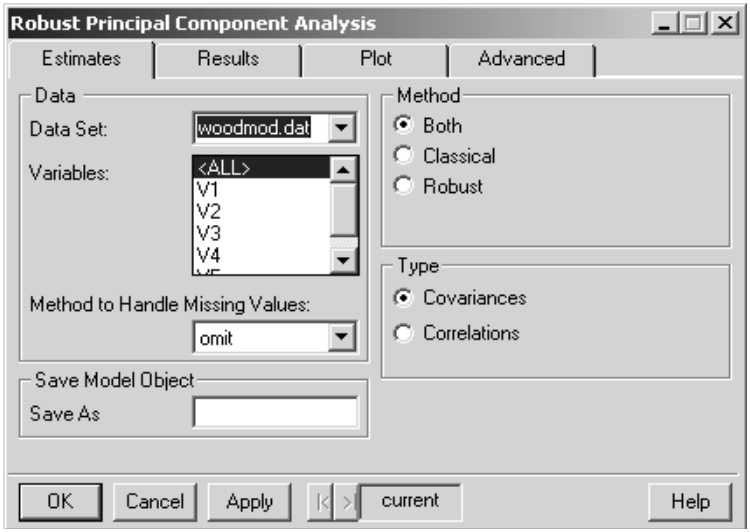


Figure 6.1: *Robust Principal Components Dialog: Estimates Page*

NOTE: You could also skip selecting `woodmod.dat` in the Object Explorer, in which case you could type `woodmod.dat` in `Data Set` combo box.

If you wish to compute the principal components for only some of the variables in the data set, then select those variables in the **Variables** list.

The default is to compute both classical and robust principal components estimates. You can compute only classical or only robust estimates by clicking the radio button of your choice in the **Method** region.

To base the principal component analysis on robust correlation estimates rather than covariance estimates, click the **Correlations** radio button in the **Type** region of the dialog.

Click on other page tabs to see what options are available for covariance estimation. For example, click the **Plot** page tab to reveal the plot options for the returned object.

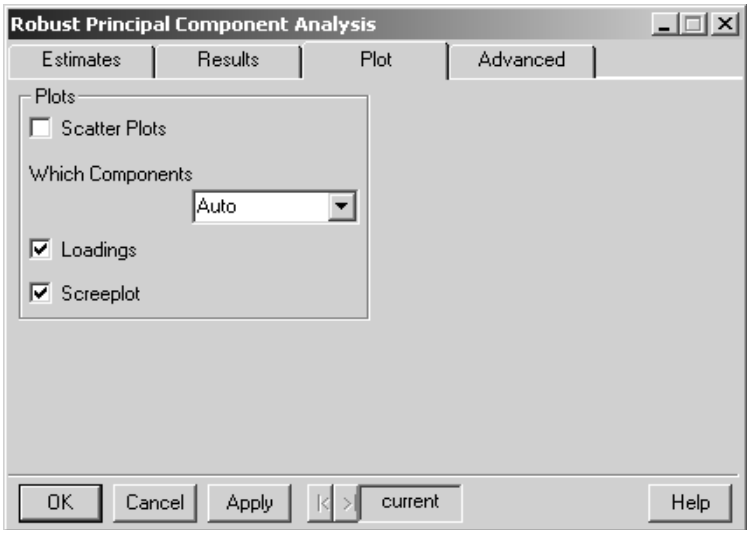


Figure 6.2: *Robust Principal Components Dialog: Plot Page*

Scatter Plot of Components

This Trellis display plots pairwise the scores of the principal components for the robust model and for the classical model. You can use the `which components` field to specify the components included in the plot. The built in choices are "`Auto`" and "`All`". "`All`" includes all of the components, and "`Auto`" displays the top five components (all

if there are less than 5). Additionally, an integer vector may also be given. For example `c(1, 2, 5)` would plot the scores for the 1^{st}, 2^{nd}, and 5^{th} components.

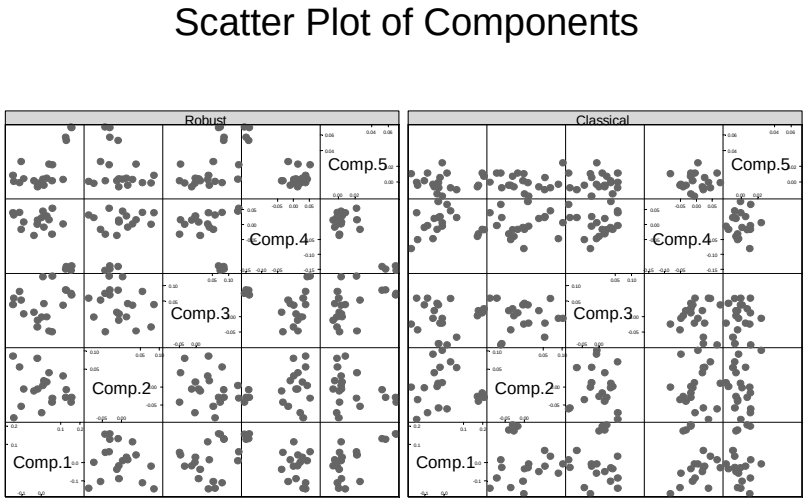


Figure 6.3: *Scatter Plot of Components*

Loadings

The second option produces a side-by-side plot of the loadings for each component. Only four components are displayed per page so that high dimensional cases will not confound the plot.

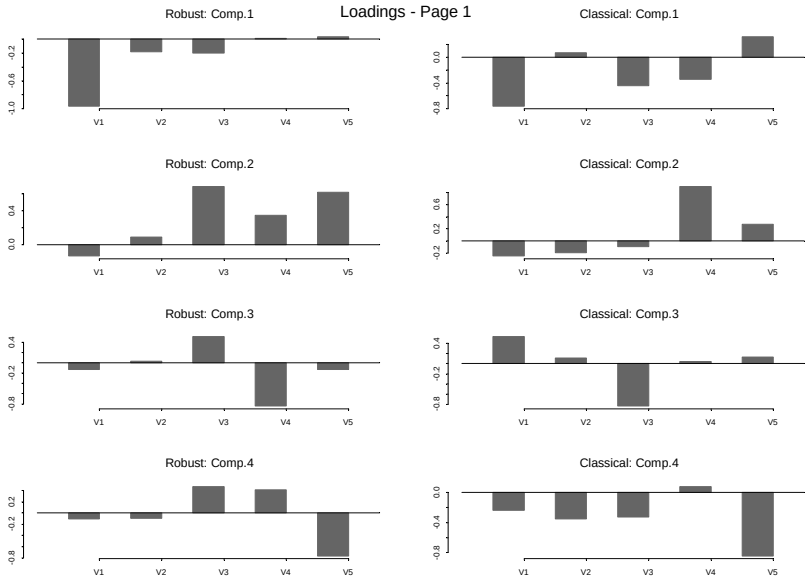


Figure 6.4: *Robust and Classical Loadings (first page only)*

Screeplot

The final choice is an overlaid screeplot.

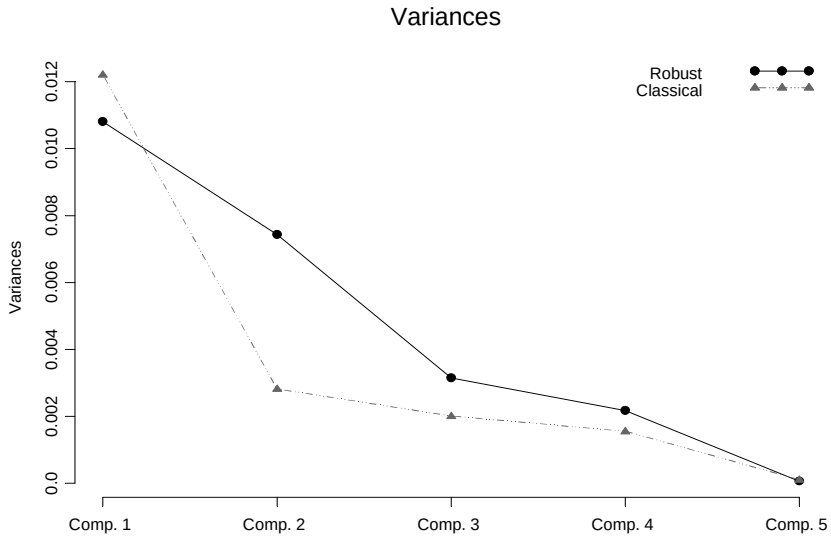


Figure 6.5: *Overlaid Screeplot*

The Statistics Report Window

The Long Output and Loadings are shown below.

```
Calls:
   Robust : princompRob(data = woodmod.dat, na.action =
na.omit)
Classical : princomp(data = woodmod.dat, na.action =
na.omit)

Importance of components:

Standard deviation
          Comp. 1 Comp. 2 Comp. 3 Comp. 4 Comp. 5
   Robust  0.1039  0.0862  0.0561  0.0466  0.0079
Classical  0.1105  0.0530  0.0448  0.0394  0.0101

Proportion of Variance
          Comp. 1 Comp. 2 Comp. 3 Comp. 4 Comp. 5
   Robust  0.4574  0.3147  0.1332  0.0920  0.0026
Classical  0.6534  0.1506  0.1074  0.0831  0.0055
```

```
Cumulative Proportion
          Comp. 1 Comp. 2 Comp. 3 Comp. 4 Comp. 5
   Robust  0.4574  0.7721  0.9054  0.9974       1
Classical  0.6534  0.8040  0.9114  0.9945       1

Loadings:
    Robust
   Comp. 1 Comp. 2 Comp. 3 Comp. 4 Comp. 5
V1 -0.962  -0.133  -0.128  -0.104  -0.172
V2 -0.181                           0.974
V3 -0.201   0.686   0.511   0.472
V4          0.347  -0.840   0.416
V5          0.619  -0.127  -0.765  -0.121

 Classical
   Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
V1 -0.760 -0.252  0.530 -0.235 -0.151
V2        -0.194  0.111 -0.349  0.907
V3 -0.444        -0.829 -0.325
V4 -0.343  0.902                0.245
V5  0.321  0.274  0.130 -0.843 -0.307
```

COMPUTING ROBUST PRINCIPAL COMPONENTS ESTIMATES AT THE COMMAND LINE

Computing Both Classical and Robust Estimates

You can compute both classical and robust estimates of the Principal Components at the command line using the **Robust Library** function `fit.models`.

```
> wood.rpc <- fit.models(list(Classical = "princomp",
+ Robust = "princompRob"), data = woodmod.dat)
> wood.rpc

Calls:
Classical : princomp(data = woodmod.dat)
   Robust : princompRob(data = woodmod.dat)

Standard deviations:
              Comp.1     Comp.2     Comp.3     Comp.4
Classical 0.11049902 0.10394430 0.05304690 0.08621876
   Robust 0.05609822 0.03940449 0.04661103 0.01010690

               Comp.5
Classical 0.044797818
   Robust 0.007907684

The number of variables is 5 and the number of observations
is 20.
```

The returned object is of class "`fit.models`". You can use the generic `plot` function on the returned object:

```
> plot(wood.rpc)
```

which displays the following menu of choices.

```
Make plot selections (or 0 to exit):

1: plot: All
2: plot: Trellis of Component Scatter Plots
3: plot: Loadings
4: plot: Variances
Selection(s):
```

Choices 2 through 4 correspond to Figure 6.3, Figure 6.4, and Figure 6.5 respectively. `All` draws all three plots in order.

The generic function **summary** is also supported for class "`fit.models`" objects. The following command will produce the output found in the Statistics Report shown above.

```
> summary(wood.rpc, loadings = T)
```

Computing Only One Estimate

Use the function `princomp` if you only want a classical principal component analysis, and use the function `princompRob` if you want only a robust estimate. For the robust case:

```
> princompRob(woodmod.dat)

Standard Deviations:
   Comp. 1    Comp. 2    Comp. 3    Comp. 4     Comp. 5
 0.1039443 0.08621876 0.05609822 0.04661103 0.007907684

The number of variables is 5 and the number of observations
is 20

Component names:

 "sdev" "loadings" "correlations" "scores" "center" "scale"
"n.obs" "call" "factor.sdev" "coef"

Call:
princompRob(x = woodmod.dat)
```

The returned object is of class "`princompRob`". The generic `plot` and `summary` functions are similar to those for the existing "princomp" class.

You can use the `princompRob` optional argument `corr=T` for a principle component analysis based on the robust correlation matrix rather than the covariance matrix:

```
> princompRob(x = woodmod.dat, corr = T)

Standard Deviations:
  Comp. 1  Comp. 2   Comp. 3   Comp. 4   Comp. 5
 1.474969 1.209257 0.8883914 0.7091129 0.2647359
```

```
The number of variables is 5 and the number of observations
is 20

Component names:

 "sdev" "loadings" "correlations" "scores" "center" "scale"
"n.obs" "call" "factor.sdev" "coef"

Call:
princompRob(x = woodmod.dat, corr = T)
```

ROBUST DISCRIMINANT ANALYSIS

7

OVERVIEW OF THE METHOD

Introduction

Suppose you have a set of quantitative observations about individuals belonging to two or more groups, along with a group identifier for each vector observation that is represented by a categorical (factor) variable. Discriminant analysis uses the quantitative variables along with the group identifier to create a model that can be used to assign new observations to one of the groups (i.e., *classify* new observations). See the documentation for the S-PLUS `discrim` function for further details about this technique.

Robust discriminant analysis is implemented for two classes of models where the quantitative data in each group follows a *nominally* multivariate normal distribution: the *homoscedastic* and the *heteroscedastic* models. We say nominally because we are allowing for heavy-tailed departures for multivariate normality that give rise to multivariate outliers.

In the homoscedastic model the covariance matrices of the groups are assumed to be the same for every group. In this case you can derive a linear discriminant function

$$l(x) = \beta_0 + \beta_1^T x$$

where β_0 is a scalar and β_1 is a vector, and you classify new observations as being in one of two groups depending on whether or not *l(x)* is larger or smaller than a certain threshold.

In the *heteroscedastic* model you do not make any assumption on the covariance matrices, and allow them to be different for each group. In this case the discriminant functions is quadratic:

$$q(x) = \beta_0 + \beta_1^T x + x^T \beta_2 x$$

where β_2 is a matrix. For classical discriminant analysis, the scalar, vector and matrix quantities β_0, β_1 and β_2 depend on maximum likelihood estimates of the multivariate group means and variance-covariance matrices μ_i and Σ_i respectively, assuming multivariate normal distributions for each group.

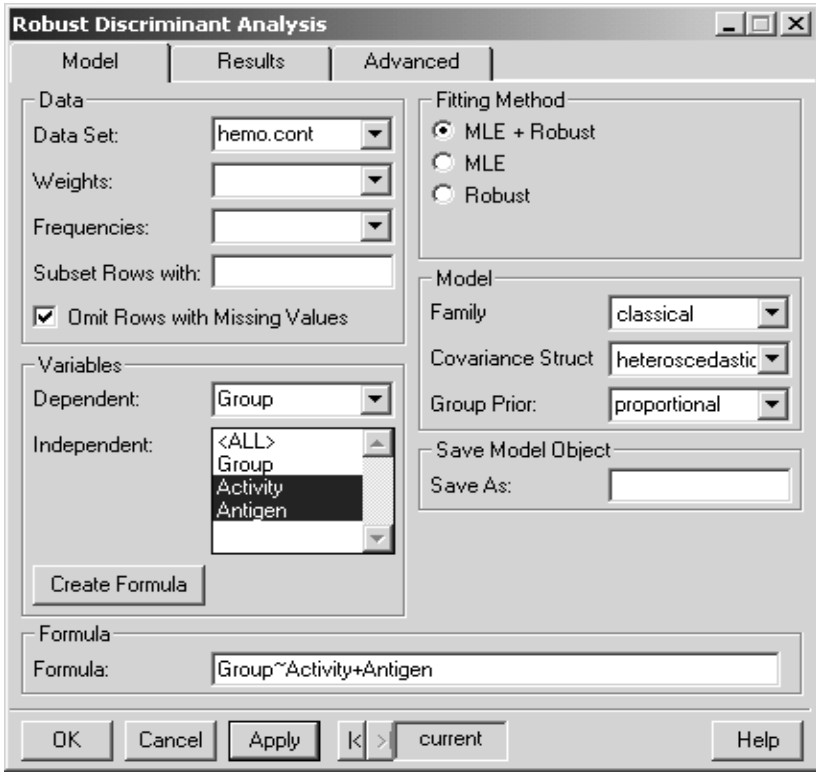
Because outliers can badly distort the Gaussian MLE's of the group means and covariances, and hence distort the linear and quadratic discriminant function, it is highly desirable to have robust discriminant methods that are not much influenced by a small fraction of outliers.

The basic idea behind the robust discriminant methods we provide you with here is to replace the Gaussian maximum likelihood estimates μ_i and Σ_i by the robust alternatives provided by the function `covRob` in this Robust Library.

COMPUTING MLE AND ROBUST DISCRIMINANT MODELS WITH THE NT/WINDOWS GUI

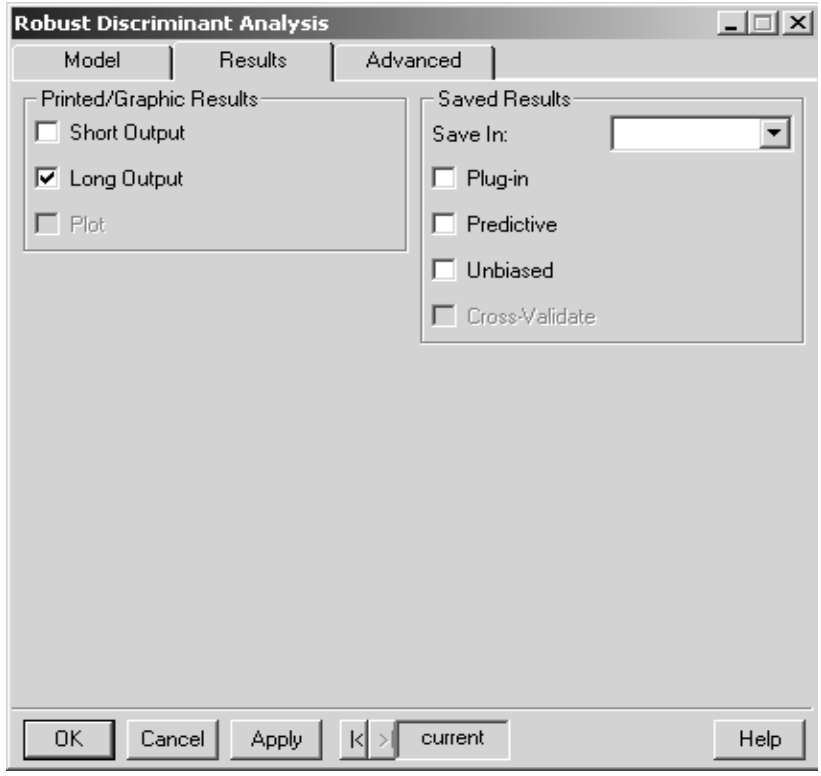
Computing Both MLE and Robust Discriminants

Consider the data frame `hemo.cont` included in the library, of observations on two variables for subjects belonging to two groups. The original data (Johnson and Wichern, 1992, page 570) has been contaminated by adding 10 outliers to the first group. Fit a discriminant analysis model to this data by choosing **Robust ► Discriminant Analysis** from the menubar. Select `Group` as the dependent variable and `Activity` and `Antigen` as your dependent variables. In the Model region, select **classical** from the **Family** pull-down menu and select **heteroscedastic** from the **Covariance Struct** pull-down menu. Save the discriminant model object by typing the name `hemo.gui` in the **Save As** box in the **Save Model Object** region.



Note that the above dialog **Model** page looks exactly like the one for Discriminant Analysis in Spotfire S+, except for the **Fitting Method** choices, with the default choice **MLE + Robust** (both maximum likelihood estimates and robust fits are computed) and alternate choices **MLE** (maximum likelihood estimates only) and **Robust** (robust fit only).

Click on the tab labeled **Results** and select the **Short Output** option only. Note that the option **Cross-Validate** that is available in the (classical) Discriminant Analysis dialog is not available here, as it demands excessive computing time for the default robust estimate implemented in `covRob`.



The Statistics Report

Click **OK**. The output comparing the MLE and the robust discriminant models appears in the **Report** window:

```
  *** Robust Discriminant Analysis ***

Calls:
   MLE : discrim(formula = structure(.Data = Group ~
Activity + Antigen, class
   = "formula"), data = hemo.cont, family = Classical(
  "homoscedastic"), na.action = na.omit, prior =
"proportional")
Robust : discRob(formula = structure(.Data = Group ~
Activity + Antigen, class
   = "formula"), data = hemo.cont, family = Classical(
  "homoscedastic"), na.action = na.omit, prior =
"proportional")

Constants:

   MLE
         1         2
 -1.120877 -1.319132

Robust
         1         2
 -1.336137 -6.527361

Linear Coefficients:

   MLE
                  1          2
Activity  0.4801456 -4.4227122
 Antigen -4.8420655 -0.7207522

Robust
                 1         2
Activity -10.50404 -35.60441
 Antigen   1.95306  12.32662
```

```
Mahalanobis Distances:

   MLE
         X1       X2
1 0.000000 2.338221
2          0.000000

Robust
         X1       X2
1 0.000000 5.843787
2          0.000000

Plug-in classification table:

   MLE
         X1 X2     Error Posterior.Error
      1 21 19 0.4750000       0.5964972
      2  0 45 0.0000000      -0.0183033
Overall       0.2235294       0.2710146

Robust
         X1 X2     Error Posterior.Error
      1 37  3 0.0750000      -0.0394930
      2  9 36 0.2000000       0.2168326
Overall       0.1411765       0.0962088
(from=rows,to=columns)

Monte Carlo Error Rates:
       Group 1 Group 2
   MLE   0.265   0.802
Robust   0.140   0.904

(conditioned on the training data)
```

Note that there is a marked difference in the values of the MLE and robust coefficient estimates for the quadratic discriminant function above. But how do you compare the relative classification error-rate performance of these two methods?

One way is to use simple "plug-in" empirical error rates for the "training" sample used to fit the discriminant model. This approach tends to give inaccurate estimates of the error rate. Another way is to simulate multivariate "test" data using the estimated means and covariances for the groups. Both of these methods are used when you check the Long Output box on the Results page of the dialog. Try this, and see what happens. You will also see these kinds of results when you carry out robust discriminant analysis from the command line as described in the next section.

NOTE: The options in the **Saved Results** region of the **Results** page are not currently available when you select **MLE+Robust** or **Robust** on the **Model** page. We expect to fix this in a future release of the Robust Library.

COMPUTING MLE AND ROBUST DISCRIMINANT MODELS AT THE COMMAND LINE

Computing Both MLE and Robust Discriminants

If you prefer to work at the command line, use the new `fit.models` function in the **Robust Library** to compute both the MLE and the robust discriminant estimates and store them in a single S-PLUS object.

```
> hemo.fit <- fit.models(model = list(MLE = 'discrim',
+ Robust = 'discRob'), formula = Group ~.,
+ data = hemo.cont, family = Classical('hetero'))
```

To obtain a detailed comparison of the fits use the `summary` method (this might take a few minutes to finish).

```
> summary(hemo.fit)

Starting simulation. Generating 1000 new random samples.
This may take a few minutes.
Done.

Starting simulation. Generating 1000 new random samples.
This may take a few minutes.
Done.

Calls:
   MLE : discrim(formula = structure(.Data = Group ~
Activity + Antigen, class = "formula"), data = hemo.cont,
family
 =
  Classical("hetero"))
Robust : discRob(formula = structure(.Data = Group ~
Activity + Antigen, class = "formula"), data = hemo.cont,
family
 =
  Classical("hetero"))

Constants:

   MLE
         1          2
 1.718957 0.04783382
```

```
Robust
        1        2
 15.06448 8.648668

Linear Coefficients:

   MLE
                  1         2
Activity -0.4619015 -21.80618
 Antigen -4.3220909  13.70132

Robust
                 1          2
Activity -11.31777 -36.418161
 Antigen   4.29550   8.399457

Quadratic Coefficients:

Group: 1

   MLE
          Activity    Antigen
Activity -4.730540  -3.213555
 Antigen -3.213555 -15.854149

Robust
          Activity   Antigen
Activity -71.99090  46.86212
 Antigen  46.86212 -52.63087

Group: 2

   MLE
          Activity   Antigen
Activity -35.85158  22.93602
 Antigen  22.93602 -35.47632

Robust
          Activity   Antigen
Activity -54.38198  14.10806
```

```
 Antigen  14.10806 -15.31899

Pairwise Generalized Squared Distances:

   MLE
         1        2
1 0.000000 15.25672
2 1.195969  0.00000

Robust
         1        2
1 0.000000 5.720328
2 8.225338 0.000000

Plug-in classification table:

   MLE
        X1 X2     Error Posterior.Error
      1 27 13 0.3250000       0.4186656
      2  1 44 0.0222222       0.0081484
Overall       0.1647059       0.2013330

Robust
        X1 X2     Error Posterior.Error
      1 31  9 0.2250000       0.0714852
      2  9 36 0.2000000       0.1097189
Overall       0.2117647       0.0917266
(from=rows,to=columns)

Monte Carlo Error Rates:
       Group 1 Group 2
   MLE   0.250   0.958
Robust   0.077   0.861

(conditioned on the training data)
```

Note the Plug-in classification table results above. The `X1` column gives the number of observations from groups 1 and 2 that are classified as group 1, in the rows labeled 1 and 2, respectively. Similarly, the `X2` column gives the number of observations that are classified as group 2.

The last table "`Monte Carlo Error Rates`" is computed as follows. We simulated 1000 observations (this number can be controlled with the option `n.MC` of `summary.discRob`, see below) for each group. These observations were drawn from a multivariate normal distribution with the mean and covariance matrix equal to the corresponding estimates. We used the discriminant function to assign them to one of the two groups, and then counted how many of these new observations were misclassified. The above table contains the corresponding proportions. Note that in this example the Robust estimates performs significantly better than the classical estimate for the first group (0.7% versus 25% misclasified observations) and both methods are comparable for Group 2 (97.5% versus 95.8%). See below to learn how to obtain such a measure for classical discriminant analysis objects of class `discrim`.

NOTE: A `predict` method does not currently exist for a `discRob` object. We expect to remedy this in a future release.

Computing only a Robust Discriminant

Use the function `discRob` to compute only the robust fit. For example:

```
> hemo.rob <- discRob(Group ~., data = hemo.cont, family =
+ Classical('hetero'))
```

By default, the `summary` method does not calculate the `Monte Carlo Error Rates` discussed above. To obtain it, use the optional arguments `MC` and `n.MC` (the number of simulated observations to generate) as follows:

```
> summary(hemo.rob, MC=T, n.MC = 500)

Starting simulation. Generating 500 new random samples.
This may take a few minutes.
Done.
```

```
Call:
discRob(structure(.Data = Group ~ Activity + Antigen,
  class = "formula"), data = hemo.cont, family =
  Classical("hetero"))



Group means:
    Activity     Antigen  N    Priors
1 -0.1237905 -0.06941429 40 0.4705882
2 -0.3465000 -0.04495833 45 0.5294118

Covariance Structure: heteroscedastic

Group: 1
           Activity    Antigen
Activity 0.01652061 0.01470982
 Antigen            0.02259764

Group: 2
           Activity    Antigen
Activity 0.01208047 0.01112554
 Antigen            0.04288533

Constants:
        1        2
 15.06448 8.648668

Linear Coefficients:
                X1        X2
Activity -11.31777 -36.41816
 Antigen   4.29550   8.39946

Quadratic coefficents:

group: 1
          Activity   Antigen
Activity -71.99090  46.86212
 Antigen           -52.63087

group: 2
```

```
          Activity   Antigen
Activity -54.38198  14.10806
 Antigen           -15.31899



Pairwise Generalized Squared Distances:
         1        2
1 0.000000 5.720328
2 8.225338 0.000000

Classification table:
        X1 X2     Error Posterior.Error
      1 31  9 0.2250000       0.0714852
      2  9 36 0.2000000       0.1097189
Overall       0.2117647       0.0917266
(from=rows,to=columns)

Monte Carlo Error Rates:
 Group 1 Group 2
    0.07    0.83
(conditioned on the training data)
```

Computing Monte Carlo Error Rates for a Classical Discriminant Model

To obtain a Monte Carlo error rates table for the classical discriminant fit you use the function `summary.discRob` on an object of class `discrim` as follows. First fit an MLE model to the data.

```
> hemo.mle <- discrim(Group ~., data = hemo.cont,
+ family = Classical('hetero'))
```

Now use the function `summary.discRob` on it.

```
> summary.discRob(hemo.mle, MC = T, n.MC = 500)

Starting simulation. Generating 500 new random samples.
This may take a few minutes.
Done.

Call:
discrim(structure(.Data = Group ~ Activity + Antigen,
```

```
  class = "formula"), data = hemo.cont, family =
  Classical("hetero"))

Group means:
     Activity      Antigen  N    Priors
1  0.05076578 -0.146597846 40 0.4705882
2 -0.30794889 -0.005988889 45 0.5294118
Covariance Structure: heteroscedastic

Group: 1
           Activity     Antigen
Activity  0.1225740 -0.02484513
 Antigen             0.03657347

Group: 2
           Activity    Antigen
Activity 0.02378340 0.01537636
 Antigen            0.02403498

Constants:
        1          2
 1.718957 0.04783382

Linear Coefficients:
                X1        X2
Activity -0.461902 -21.80618
 Antigen -4.322091  13.70132

Quadratic coefficents:

group: 1
          Activity   Antigen
Activity -4.730540  -3.21356
 Antigen           -15.85415

group: 2
          Activity   Antigen
Activity -35.85158  22.93602
 Antigen           -35.47632

Pairwise Generalized Squared Distances:
```

```
          1        2
1 0.000000 15.25672
2 1.195969  0.00000
```

```
Plug-in classification table:
        X1 X2     Error Posterior.Error
      1 27 13 0.3250000       0.4186656
      2  1 44 0.0222222       0.0081484
Overall       0.1647059       0.2013330
(from=rows,to=columns)

Monte Carlo Error Rates:
 Group 1 Group 2
   0.244   0.942
(conditioned on the training data)
```

ROBUST ASYMMETRIC DISTRIBUTION PARAMETER ESTIMATION

8

Tuning constants 176

OVERVIEW OF THE METHOD

Introduction

Asymmetric distributions of positive random variables occur in many statistical applications concerning, for example, survival data (medicine), yield data (agriculture, industry), failure times (industry), income and resource consumption measures (econometric studies).

We focus on two particular aspects of this kind of data. First, the population mean or the total of a finite population is the characteristic of interest; because the total is a multiple of the mean, we concentrate on the population mean. Second, the data may contain values that are markedly different from most others. When some of these values are observed, the sample mean can be much larger than when none are observed, and, therefore, it varies strongly from sample to sample.

Transformations are often used to make the distribution (closely) symmetric; many robust procedures based on the Gaussian model are then available to estimate the transformed mean. Unfortunately, symmetrizing transformations do not always exist. Moreover, the original mean cannot usually be estimated by transforming back the transformed mean. For example, to estimate a lognormal mean both the estimates of normal mean and scale are required; therefore, the distinction between main (mean) and nuisance (scale) parameters–that characterizes most location and scale procedures–is not appropriate anymore. The Robust Library makes available two procedures that do not make this distinction: the truncated mean estimate proposed by Marazzi and Ruffieux (1999), and the B_s^p-estimate proposed in Hampel *et al.* (1986, pp 238-257).

Function Names

Table 8.1: *Function Names*

	Classical	Robust
Gamma	gammaMLE	gammaRob
Weibull	weibullMLE	weibullRob
Lognormal	lognormMLE	lognormRob

COMPUTING MLE AND ROBUST FITS WITH THE NT/ WINDOWS GUI

Computing Both MLE and Robust Fits

Consider the data `los` included in this library. The vector `los` represents the observed lengths of stay (LOS) in days of 32 patients hospitalized in a Swiss hospital during 1988 for certain "disorders of the nervous system" (Marazzi, Paccaud and Ruffieux, 1998). LOS is an important indicator of hospital costs. The LOS means of medically homogeneous groups of patients (e.g., the group considered here) within a hospital are used for hospital planning and budgeting; the means of different hospitals are used to compare costs and explore possible reductions.

The mean is 25.5 days. In many applications it is useful to use a mathematical model in order to summarize the entire distribution; for example, one can use the Gamma distribution to model and simulate hospital stays. To adjust a Gamma distribution according to both the maximum likelihood criterion and the robust estimate use the GUI dialog as follows. Choose **Robust ► Asymmetric Parameter Estimation.** The dialog window is shown in Figure 8.1.

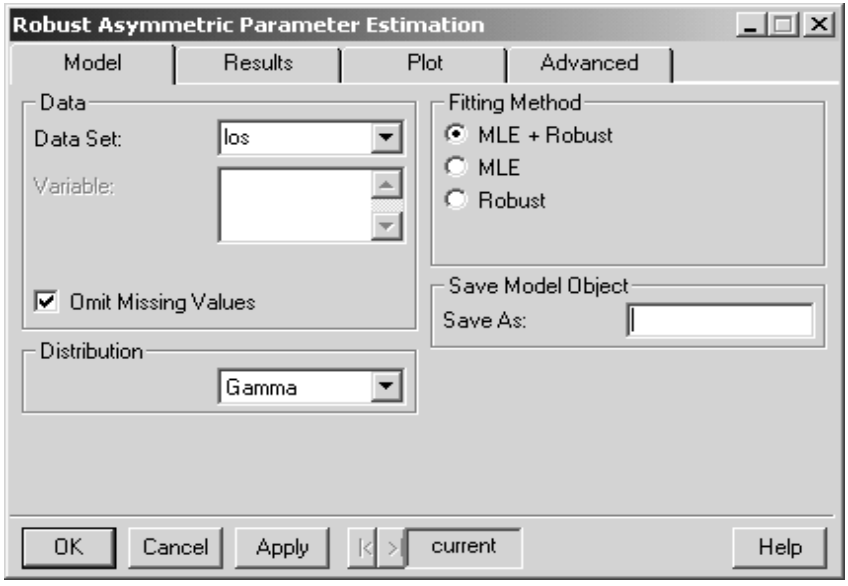


Figure 8.1: *The Robust Gamma and Weibull Fit Dialog*

When you press **OK** the results appear in a **Report** window and **a Graph Sheet.** The plot is shown in Figure 8.2.

Diagnostic Plots

In the **Graph Sheet** you obtain a histogram of the data with the overlaid estimated density functions (one for the MLE estimate and one for the Robust estimate) and a QQ plot showing the response vs. estimated quantiles. These plots appear in Figure 8.2 and Figure 8.3.

The Statistics Report

In the **Report** window you obtain a short description of the estimates (or a longer more detailed listing if you check the **Long output** in the **Results** page of the GUI Dialog).

COMPUTING MLE AND ROBUST ESTIMATES AT THE COMMAND LINE

Computing Both MLE and Robust Fits

To fit the same models using the command line use the **Robust Library** function **fit.models**.

```
> los.fits <- fit.models(list(mle = 'gammaMLE',
+ robust = 'gammaRob'), data = los)
```

The detailed summary of the estimates obtained by checking the **Long Output** option on the **Results** tab page of the GUI Dialog is obtained using the generic summary method on the `fit.models` object (los.fits).

```
> summary(los.fits)

Calls:
   mle : gammaMLE(data = los)
robust : gammaRob(data = los)

Coefficients:
               Estimates Std. Error
   mle : Alpha 0.9263    0.7735
robust : Alpha 1.3858
   mle : Sigma 8.4992    0.0645
robust : Sigma 3.6396
   mle :    Mu 7.8730    0.4609
robust :    Mu 4.9727    0.2750
```

Diagnostic Plots

The generic plot method is also implemented for fit.models objects.

```
> plot(los.fits)

Make plot selections (or 0 to exit):

1: plot: All
2: plot: Overlaid Density Estimates
3: plot: Response vs Estimated Quantiles
Selection(s): 1
```

Select 1 to produce the following diagnostic plots.

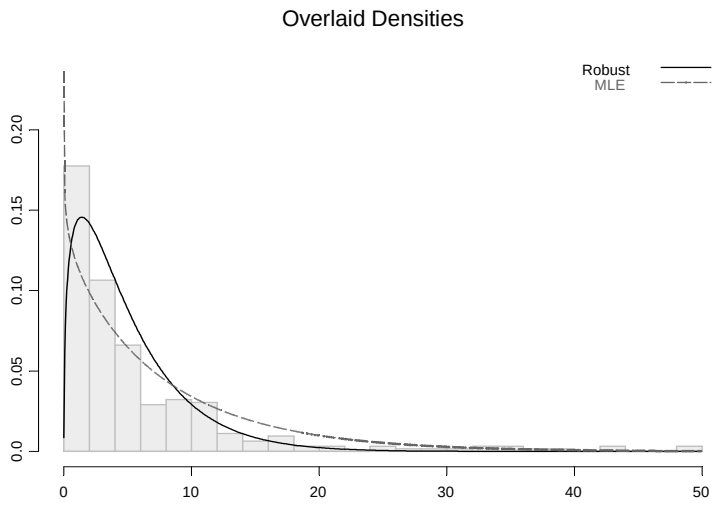


Figure 8.2: *MLE and Robust Gamma fits for the* `los` *data*

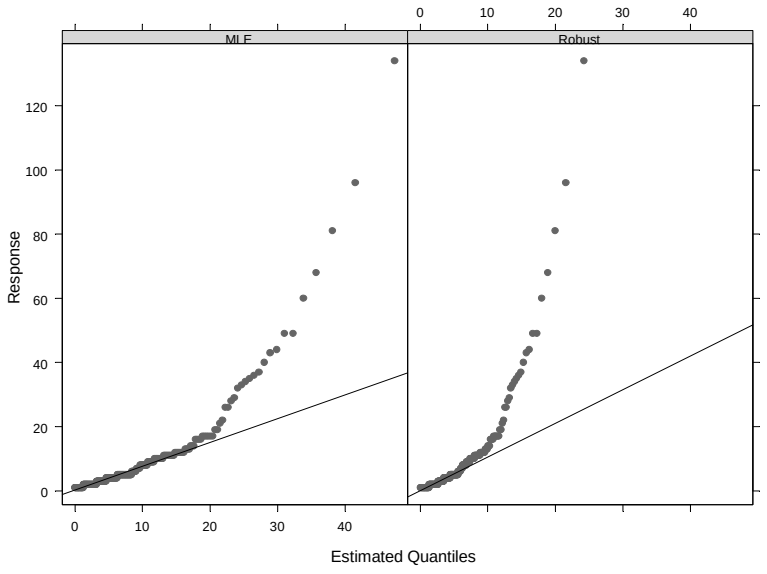


Figure 8.3: *Response vs. Estimated Quantiles for both the MLE and Robust fits*

Note that the robust estimate has been applied to the entire data set but that the Gamma distribution has automatically been fitted to the "majority" of the data. The mean of the fitted model is 4.97 days.

Computing only a Robust GLM Fit

If you want to fit a robust estimate to the `los` data you can use the function `gammaRob()`

```
> los.robust.fit <- gammaRob(los)
```

Detailed information on the estimated parameters is obtained by using the `summary()` function.

```
> summary(los.robust.fit)

Robust gamma distribution parameter estimate

Call:
  gammaRob(data = los)

Coefficients:
      Estimates Std. Error
Alpha 1.3857686
Sigma 3.6395658
   Mu 4.9726962 0.2749752
```

The Diagnostic Plots

The command

```
> plot(los.robust.fit)
```

is used to produce the plots similar to those in Figure 8.2 and Figure 8.3 except that only the information pertaining to the robust fit is displayed.

CONTROLLING OPTIONS FOR ROBUST WEIBULL AND GAMMA FITS

The parameters that control the computational procedures used to compute the MLE and robust estimates for both gamma, lognormal, and Weibull distributions are passed to their respective functions through the control argument. The **Robust Library** provides the following funcions which are useful for creating these lists.

- `gammaMLE.control`
- `gammaRob.control`
- `lognormRob.control`
- `weibullMLE.control`
- `weibullRob.control`

The functions for generating the control lists for the robust estimators each have one required argment: the name of the estimator (the same value that is passed to the `estim` argument in `gammaRob`, `lognormRob`, or `weibullRob`). If no additional arguments are provided then a list with the control parameters each set to their default valu is returned. Only the parameters required for the specified estimator are returned. Please refer to the online help for explanations of the parameters and their default settings. For example,

```
> help(lognormRob.control)
```

will display the help for lognormRob.control. Also, it may be helpful to note that the control list can be save in your working chapter and used for several analyses. For example, we can shorten th initial interval used by the truncated mean estimator:

```
> los.robust.control <- gammaRob.control("tdmean",
+ alpha1 = .5, alpha2 = 10.5)
> los.robust.fit <- gammaRob(los, robust.control =
+ los.robust.control)
```

Note that control arguments can also be passed directly to the corresponding function. Internally they are processed through the corresponding control function. Hence the following is equivalent to the lines above.

```
> los.robust.fit <- gammaRob(los, alpha1 = .5,
+ alpha2 = 10.5)
```

The same applies to all the asymmetric distribution parameter estimation functions included in the Robust Library.

THEORETICAL DETAILS

The Models

Let $Y>0$ be a random variable with unknown cumulative distribution function G and asymmetric density g. We suppose that, as it often occurs in practice, we are mainly interested in estimating $\mu(G)$, the expected value of Y, using n independent observations $x_1, \ldots, x_n$. Moreover, we are willing to use a parametric model $F_{\alpha,\sigma}$ for the distribution $F = G \circ h^{-1}$ (with density f) of some monotone increasing transformation $Y = h(X)$. The corresponding model for G is $G_{\alpha,\sigma}$ (density $g_{\alpha,\sigma}$) and define $\mu = \mu(G_{\alpha,\sigma})$. We assume that: (a) σ is a scale parameter of $F_{\alpha,\sigma}$; and (b) either α is a location parameter or α is a shape parameter of $F_{\alpha,\sigma}$.

Two popular asymmetric models that satisfy (a) and (b) are the Weibull and the Gamma distributions. The *Weibull distribution* with shape τ and scale v has density

$$g_{\tau,v}(x) = (\tau/v)(x/v)^{\tau-1}\exp(-(x/v)^{\tau}), \quad x>0, \quad v>0, \quad \tau>0.$$

The mean is $v\Gamma(1+1/\tau)$. It is often convenient to consider $Y = \ln(X)$; the density of Y is then

$$f_{\alpha,\sigma}(x) = \exp([((y-\alpha)/\sigma) - \exp((y-\alpha)/\sigma)]/\sigma).$$

where $\alpha = \ln v$ is a location parameter and $\sigma = 1/\tau$ a scale parameter. The *Gamma distribution* with shape α and scale σ has density

$$f_{\alpha,\sigma}(y) = (\sigma\Gamma(\alpha))^{-1}(y/\sigma)^{\alpha-1}\exp(-y/\sigma), \quad y>0, \ \sigma>0, \ \alpha>0.$$

We have $\mu(G_{\alpha,\sigma}) = \sigma\alpha$.

The B_s^p-estimate

This estimator is used to estimate the gamma and Weibull distribution parameters. Let $S(y, \theta)$ denote the score vector function of F_θ, i.e., $S(y, \theta) = \partial \ln f_\theta(y) / \partial\theta$ and let b be a user chosen tuning constant. The B_s^p-estimate $\hat{\theta}$ of θ is then defined as a solution of

$$\sum_{i=1}^{n} \Psi_{\underline{b}}[A_{\underline{b}}(\theta)(S(y, \theta) - C_{\underline{b}}(\theta))] = 0,$$

where the function $\Psi_{\underline{b}}(z)$, $z = (z_1, z_2) \in \Re^2$ is defined by $\Psi_{\underline{b}}(z) = (H_{b_1}(z_1), H_{b_2}(z_2))^t$, $\underline{b} = (b_1, b_2) \in \Re^2$ and $H_b(z) = min(b, max(y, -b))$ denotes the Huber function. Moreover, $A_{\underline{b}}(\theta)$ is a 2 by 2 non-singular lower triangular matrix and $C_{\underline{b}}(\theta)$ is a 2-component vector; they are both functions of θ and are defined jointly and implicitly by the following equations

$$\int \Psi_{\underline{b}}[A_{\underline{b}}(\theta)(S(y, \theta) - C_{\underline{b}}(\theta))]\Psi_{\underline{b}}[A_{\underline{b}}(\theta)(S(y, \theta) - C_{\underline{b}}(\theta))]^t f_\theta(y)\partial y = I$$

$$\int \Psi_{\underline{b}}[A_{\underline{b}}(\theta)(S(y, \theta) - C_{\underline{b}}(\theta))]f_\theta(y)\partial y = 0$$

(see Hampel *et al.*, 1986, pp 238-256). The corresponding estimate of μ is $\hat{\mu} = \mu(G_{\hat{\theta}})$.

Under certain regularity conditions $\hat{\theta}$ is asymptotically normally distributed with asymptotic covariance matrix

$$V(\hat{\theta}, F) = M(\psi, F)^{-1} Q(\psi, F) M(\psi, F)^{-t},$$

where

$$Q(\psi, F) = \int \psi(y, \hat{\theta})(\psi(y, \hat{\theta}))^t \partial F(y) ,$$

$$M(\psi, F) = -\int \left[\frac{\partial}{\partial\theta}\psi(y, \hat{\theta})\right]_{\theta = \hat{\theta}} \partial F(y) .$$

Let F_n denote the empirical distribution of the sample. In practice we use $V(\hat{\theta}, F_n)/n$ or $V(\hat{\theta}, F_{\hat{\theta}})/n$ as approximations for the covariance matrix of $\hat{\theta}$.

Computations

The computation of the Bsp-estimate is discussed in Marazzi and Ruffieux (1996); details can be found in Marazzi and Randriamiharisoa (1997a, 1997b, 1997c). The general approach is as follows: given an initial value for θ, one computes $A_{\underline{b}}(\theta)$, $C_{\underline{b}}(\theta)$ and an improved value of θ. However, in the Gamma case, $A_{\underline{b}}(\theta)$ and $C_{\underline{b}}(\theta)$ depend on α. The repeated computation of $A_{\underline{b}}(\theta)$ and $C_{\underline{b}}(\theta)$ slows down iterations. Therefore, the following two-step procedure is used: (a) for fixed b, determine $A_{\underline{b}}(\theta)$ and $C_{\underline{b}}(\theta)$ for a discrete set of values of α; (b) solve the equations for θ using a linear interpolation of the tables obtained in step (a) in order to compute $A_{\underline{b}}(\alpha)$ and $C_{\underline{b}}(\alpha)$. This scheme is particularly time efficient when the estimator has to be evaluated for a sequence of similar problems, such as those required in a bootstrap procedure.

The Truncated Mean Estimate

This estimator is used to estimate the lognormal distribution parameters. Let $m(F_n)$ and $s(F_n)$ be robust measures of location and dispersion based on n (transformed) observations $y_1,...,y_n$. Specifically, we take the median and the median absolute deviation or the β-trimmed mean and the γ-trimmed absolute deviation defined below. Let $m(F)$ and $s(F)$ denote the asymptotic values of $m(F_n)$ and $s(F_n)$ when the data is distributed according to the distribution F. The truncated mean is based on four steps. First, an initial estimate $(\tilde{\alpha}, \tilde{\sigma})$ is computed by solving

$$m(F_{\tilde{\alpha},\tilde{\sigma}}) = m(F_n) \text{ and } s(F_{\tilde{\alpha},\tilde{\sigma}}) = s(F_n)$$

Second, an upper truncation limit T_u is defined as the u-quantile $T_u = G^{-1}_{\tilde{\alpha},\tilde{\sigma}}(u)$ of the fitted model for X, where u is a user chosen number (i.e., a "tuning constant") e.g., $u = 0.99$. Third, a lower limit T_l is determined so that the mean of the truncated model coincides with the mean of the entire model, i.e.,

$$\frac{1}{u - G^{-1}_{\tilde{\alpha},\tilde{\sigma}}(T_l)} \int_{T_l}^{T_u} x g_{\tilde{\alpha},\tilde{\sigma}}(x)dx = \int x g_{\tilde{\alpha},\tilde{\sigma}}(x)dx$$

The truncated mean estimate $\bar{\mu}$ is the arithmetic mean of the x_i such that $T_l < x_i \leq T_u$:

$$\bar{\mu} = \text{ave}\{x_i | T_l < x_i < T_u\}.$$

The truncated mean estimate does not strongly depend on the parametric model which is only used to compute T_l and T_u. It is very robust because it completely rejects extreme observations (its breakdown point is the minimum between the breakdown points of s and m). The influence functions $IF(y;((\tilde{\alpha}, \tilde{\sigma}), F))$ and $IF(y;\bar{\mu}, G)$ can be computed but the formulae are cumbersome (Marrazzi and Ruffieux, 1999). The asymptotic variance of $\bar{\mu}$ is obtained using,

$$V(T, F) = \int IF(y, T, F)IF(y, T, F)^{\mathrm{T}} dF(y).$$

The efficiency of the truncated mean with respect to the maximum likelihood estimator depends on u (see tuning constants) and can be very high.

Trimmed Mean and Trimmed Absolute Deviation

The β-trimmed mean $m(F_n)$ and the γ-trimmed absolute deviation $s(F_n)$ of $y_1,...,y_n$ are defined as follows:

$$m(F_n) = \frac{1}{i_u + r_u - i_l - r_l} \sum_{i=i_l+2}^{i_u} y_{[i]} + r_u y_{[i_u+1]} + (1-r_l) y_{[i_l+1]},$$

$$d(F_n) = \frac{1}{j_u + s_u - j_l - s_l} \sum_{j=j_l+2}^{j_u} z_{[j]} + s_u z_{[j_u+1]} + (1-s_l) z_{[j_l+1]},$$

where $i_u = \lfloor n(1-\beta) \rfloor$, $i_l = \lfloor n\beta \rfloor$, $r_u = n(1-\beta) - i_u$, $r_l = n\beta - i_l$, $j_u = \lfloor n(1-\gamma) \rfloor$, $j_l = \lfloor n\gamma \rfloor$, $s_u = n(1-\gamma) - i_u$, $s_l = n\gamma - i_l$, $z_i = |y_i - m(F_n)|$; moreover, $y_{[1]} \le y_{[2]} \le .. \le y_{[n]}$ denote the order statistics and the notation $\lfloor z \rfloor$ is used for the largest integer smaller than z. The constants $\beta \in [0,\ 0.5]$ and $\gamma \in [0,\ 0.5]$ are user chosen. For $\beta = \gamma = 0.5$, m is the median and d is the median absolute deviation. For β and γ close to 0.5, m and d are "smoothed" versions of these estimates. By default we use $\beta = \gamma = 0.4$.

Tuning constants

Classical statistics assumes that Y is distributed according to some distribution F_θ for an unknown value of a parameter $\theta \in \Re^p$. The maximum likelihood (ML) criterion is often used to estimate this value of θ. In robustness theory, one assumes that F belongs to a neighborhood P of one of the F_θ, say F_{θ_*}. For example, one can use the ε-contamination model (Huber, 1981)

$$F \in P_\varepsilon = \{G \mid G = (1-\varepsilon) F_{\theta_*} + \varepsilon H,\ H \text{ arbitrary}\}.$$

If we let μ_{MLE} denote the Maximum Likelihood (ML) estimate of μ, then the asymptotic relative efficiency (ARE) of μ with respect to μ_{MLE} at the distribution F is

$$ARE(\hat{\mu}, \hat{\mu}_{MLE}, F) = (V(\hat{\mu}_{MLE}, F) / V(\hat{\mu}, F)).$$

ARE is usually evaluated at the model F_θ. We also define the maximum asymptotic variance (MAV) of μ over the neighborhood P_ε of distribution functions, for a given mean μ^*, as

$$MAV(\hat{\mu}, \mu^*, \varepsilon) = \sup_{G \in P_\varepsilon} V(\hat{\mu}, G) .$$

The most common rule for determining the tuning constants of an M-estimator is to require that the ARE with respect to the ML estimator at the model be a certain value, e.g., 90%. The higher the value of ARE, the more the estimate is sensitive to outliers. There are, however, several pairs of values of (b_1, b_2) with the same ARE, and the rule does not determine them uniquely. As a remedy, it has been proposed to minimize MAV as a function of $\underline{b}$ and find optimal values $\underline{b}(\varepsilon)$ for varying ε and α. The sensitivity of the estimate to contamination depends then on ε and on the optimal value $\underline{b}(\varepsilon)$ of $\underline{b}$. The choice of ε would be made on the grounds of collateral information about the frequency of outliers. It turns out, however, that minimization of MAV under the constraint $b_1 = b_2$ gives almost the same minimum as the unconstrained minimization. Therefore, we suggest to use a single tuning constant $b = b_1 = b_2$. Unfortunately, the ARE of the mean estimate depends on the estimated value of the parameters and the choice of the tuning constant must be made on the grounds of a preliminary guess of θ. In the Weibull and Gamma

cases ARE only depends on α and hence, given a rough estimate for α, Table 1 allows us to choose b to obtain a certain ARE at the model.

ARE		**Gamma**	**Weibull**	**Lognormal**
	α	***b***	***b***	***u***
0.85	1	1.76	1.83	.997
	2	1.48	1.38	.996
	3	1.39	1.26	.995
	4	1.35	1.25	.993
	5	1.33	1.22	.992
	10	1.29	1.21	.991
0.90	1	2.12	2.18	-
	2	1.76	1.67	.998
	3	1.62	1.39	.997
	4	1.55	1.36	.996
	5	1.51	1.34	.996
	10	1.44	1.32	.995

CONTRIBUTED CODE

9

OVERVIEW OF CONTRIBUTED CODE

This chapter very briefly describes contributed code for three statistical methods: (1) Robust smoothing splines, (2) A robust version of Mallows' Cp for robust selection of linear models, and (3) Quantile regression. The methodology and S-PLUS functions for the robust smoothing splines and robust Cp were developed by combinations of the following individuals: Eva Cantoni, Evezio Ronchetti, Suzanne Sommer, and Robert Staudte. The regression quantiles methodology and S-PLUS code were developed by Roger Koenker.

We did not have time to develop a menu/dialog interface for these S-PLUS functions, so they are only available at the command line. Also, we only had time for minimal testing of the S-PLUS functions. None-the-less, we hope they will be of interest to users of the Robust Library, and that user feedback and interest will result in further development of these functions as needed.Computing Robust Covariance at the Command Line

Robust Smoothing Splines

You fit a robust smoothing spline that is not distorted by outliers with the function `smooth.spline.Rob`. Here is a brief example:

```
> attach(ethanol)
> plot(E, NOx)
> temp.cv <- smooth.splineRob(E, NOx, lambda = "cv")
> lines(temp.cv)
```

The resulting plot is shown in the figure below.

For further details see the Help file for `smooth.splineRob`. See also, Cantoni and Ronchetti (2000), "Resistant selection of the smoothing parameter for smoothing splines", *Statistics and Computing*.

NOTE: The function `smooth.spline.Rob` bundles up several functions provided by Eva Cantoni and Elvezio Ronchetti, namely:

`e.psi, frob, matS, my.smooth.spline, opt.RCp, opt.cv, psihuber`.

Although we do not provide Help files for these individual functions, you may view them by printing `smooth.splineRob`.

.

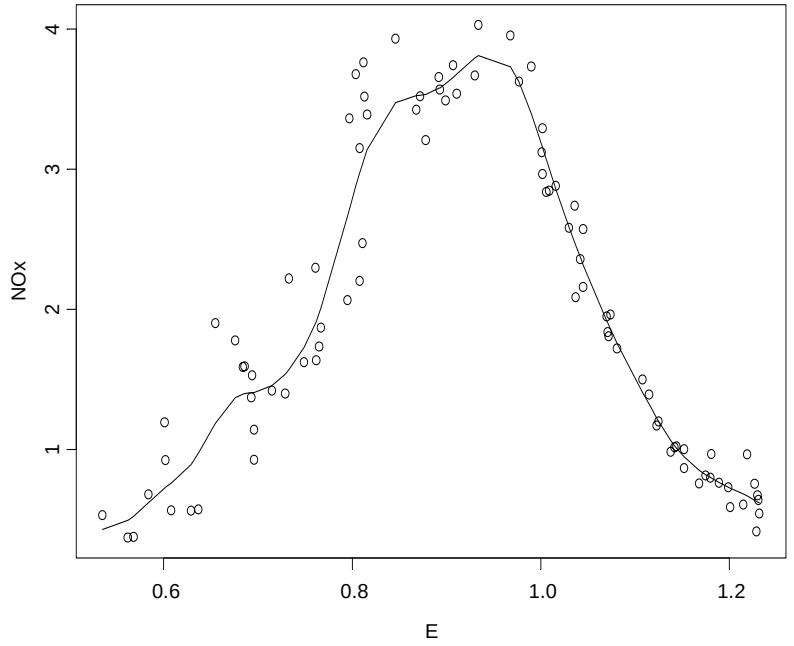


Figure 9.1: *Robust Smoothing Spline Fit*

Robust Cp

The function `RCp` computes a robust version of Mallows' Cp based on weighted least squares, which provides for robust model selection of subset models that fit the bulk of the data in the presence of outliers.

Here is a simple example of using `RCp`:

```
> rcp.result <- RCp(stack.x, stack.loss)
> plot(rcp.result)
```

This results in the plot shown below. You can also get a long summary with:

```
> summary(rcp.result)
```

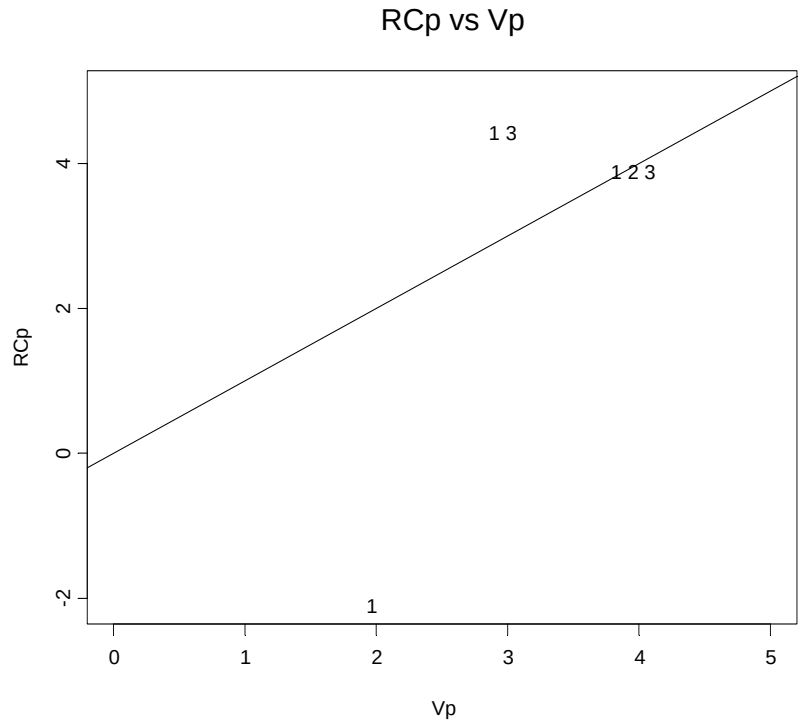


Figure 9.2: *Robust Cp Plot for Stack Loss Data*

For further details see the Help file for `RCp`. See also, Ronchetti and Staudte (1994), "A robust version of Mallows' Cp", *Jour. Amer. Statist. Assoc., 89*, 550-559, and Sommer and Staudte (1995), "Robust variable selection in regression in the presence of outliers and leverage points", *Australian Jour. of Statistics, 37,* 323-336..

NOTE: The function `RCp` bundles up several functions provided by Eva Cantoni and Elvezio Ronchetti, namely:

```
Bfinal, Hwt, Label, Mwt, plot.RCp, Plot.cp, RCp, RCp.reduced,
select.best, Subset.
```

Although we do not provide Help files for these individual functions, you may view them by printing `RCp`.

Quantile Regression

The function `rq` computes a quantile regression fit. This methodology has been extensively developed by Roger Koenker. See for example Koenker and Bassett (1978), "Regression quantiles", *Econometrica*, 46, 33-50, Koenker and d'Orey (1987, 1994), and "Computing regression quantiles", *Applied Statistics, 36*, 383-393, and *43,* 410-414.

For example:

```
> qreg.mod <- rq(stack.loss ~ stack.x, .4)
```

computes a .4 quantile regression for the stack loss data.

See the Help file for `rq` for more examples, further use details, and additional references.

REFERENCES

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

Breslow, N. E. (1996). Generalized Linear Models: Checking Assumptions and Strengthening Conclusions. *Statistica Applicata*, 8, 23-41.

Carroll, R. J., and Pederson, S. (1993). On Robustness in the Logistic Regression Model, *Journal of the Royal Statistical Society, B,* **55**, 693-706.

Copas, J. B. (1988). Binary Regression Models for Contaminated Data, *Journal of the Royal Statistical Society, B,* **50**, 225-265.

Dodge, Y. (1996). The Guinea Pig of Multiple Regression. In Rieder (ed.), *Robust Statistics, Data Analysis, and Computer Intensive Methods*. Springer, New York.

Garcia Ben, M., and Yohai, V. J. (2000). Quantile-quantile plot for the deviance residuals in the Generalized Linear Model. Technical Report, Department of Mathematics, FCEN, University of Buenos Aires.

Gervini, D., and Yohai, V. J. (1999). A Class of Robust and Fully Efficient Regression Estimates. mimeo, Universidad de Buenos Aires.

Hampel, F., Ronchetti, E. M., Rousseeuw, P. J., and Stahel, W. A. (1986). *Robust Statistics: the Approach Based on Influence Functions*. New York: John Wiley & Sons, Inc.

Hawkins, D. M., Bradu, D., and Kass, G. V. (1984). Location of Several Outliers in Multiple Regression Data Using Element Sets. *Technometrics*, 26, 197-208.

Huber, P. J. (1981). *Robust Statistics*. New York: John Wiley & Sons, Inc.

Hubert, M., and Rousseeuw, P. J. (1997). Robust Regression with Both Continuous and Binary Regressors. *Journal of Statistical Planning and Inference*, 57, 153-163.

Künsch, H. R., Stefanski, L. A., and Carroll, R. J. (1989). Conditionally Unbiased Bounded-Influence Estimation in General Regression Models, with Applications to Generalized Linear Models. *Journal of the American Statistical Association*, 84, 460-466.

Lawson, J., and Gold, L. (1988). Robust Estimation Techniques for Use in Analysis of Unreplicated 2^k and 2^{k-p} Designs.

Marazzi, A. (1993). *Algorithms, Routines, and S Functions for Robust Statistics*. Pacific Grove, CA: Wadsworth & Brooks/Cole.

Marazzi, A., Paccaud F., Ruffieux, C., and Beguin, C. (1998). Fitting the distributions of length of stay by parametric models. *Medical Care*, **36**, 915-927.

Marazzi, A., Randriamiharisoa A., (1997a). S-PLUS functions for robust estimators of the parameters of the gaussian and the lognormal distributions. Technical report available at www.hospvd.ch/iumsp.

Marazzi, A., Randriamiharisoa A., (1997b). S-PLUS functions for robust estimators of the parameters of the gamma distribution. Technical report available at www.hospvd.ch/iumsp.

Marazzi, A., Randriamiharisoa A., (1997b). S-PLUS functions for robust estimators of the parameters of the Weibull distribution. Technical report available at www.hospvd.ch/iumsp.

Marazzi, A., Ruffieux C. (1996). Implementing M-estimators of the Gamma distribution. In H. Rieder (Ed.), *Robust statistics, data analysis, and computer intensive methods, In honor of Peter Huber's 60th birthday*, Lecture Notes in Statistics, 109, Springer Verlag, Heidelberg.

Marazzi A., Ruffieux C. (1999). The truncated mean of an asymmetric distribution. *Computational Statististic and Data Analysis*, 32, pp. 79-100.

Maronna, R. A., and Yohai, V. J. (1995). The Behavior of the Stahel-Donoho Robust Multivariate Estimator. *Journal of American Statistical Association*, 90, 330-341.

Maronna, R. A., and Yohai, V. J. (1999). Robust Regression with Both Continuous and Categorical Predictors. mimeo, Universidad de Buenos Aires.

Maronna, R. A., and Zamar, R. H. (2001). Robust Multivariate Estimators for High Dimensional Data Sets, pre-print.

Martin, R. D., and Zamar, R. H. (1989). Asymptotically Min-Max Robust M-estimates of Scale for Positive Random Variables. *Journal of the American Statistical Association*, 84, 494-501.

Martin, R. D., and Zamar, R. H. (1993). Bias Robust Estimates of Scale. *Annals of Statistics*, 21: 991-1017.

Pena, D., and Yohai, V. (1999). A Fast Procedure for Outlier Diagnostics in Large Regression Problems, *Journal of the American Statistical Association*, 94, 434-445.

Pregibon, D. (1981). Logistic Regression Diagnostics. *Annals of Statistics*, 9, 705-724.

Rocke, D. M. (1996). Robustness Properties of S-estimators of Multivariate Location and Shape in High Dimension. *Annals of Statistics*, vol. 24, no. 3, 1327-1345.

Ronchetti, E. (1985). Robust Model Selection in Regression. *Statistics & Probability Letters*, 3, 21-23.

Rousseeuw, P. J. (1984). Least Median of Squares Regression. *Journal of the American Statistical Association*, 79, 871-881.

Rousseeuw, P. J., and Leroy, A. M. (1987). *Robust Regression and Outlier Detection.* New York: Wiley.

Rousseeuw, Peter, and Van Driessen, Katrien (1999). The algorithm fast-mcd for robust estimation. *Wiskundig bekeken: liber amicorum Roger De Groot* / van Wouwe Martine [edit.], e.a., Gent, Mys & Breesch, p. 163-168.

Rousseeuw, P. J., and van Zomeren, B. C. (1990). Unmasking Multivariate Outliers and Leverage Points. *Journal of the American Statistical Association*, 85, 633-639.

Rousseeuw, P. J., and Yohai, V. (1984). Robust Regression by Means of S-estimators. In *Robust and Nonlinear Time Series Analysis*, J. Franke, W. Hardle, & R. D. Martin (Eds.), Lecture Notes in Statistics, 26, 256-272, Springer-Verlag.

Stefanski, L. A., Carroll, R. J., and Ruppert, D. (1986). Optimally Bounded Score Functions for Generalized Linear Models with Applications to Logistic Regression. *Biometrika*, 73, 413-425.

Wagner, J. (1994). Regionale Beschäftigungsdynamik und höherwertige Produktionsdienste: Ergebnisse fur den Grossraum Hannover (1979-1992). *Raumforschung und Raumordnung*, 52, 146-150.

Yohai, V. J. (1987). High Breakdown-Point and High Efficiency Estimates for Regression. *Annals of Statistics*, 15, 642-665.

Yohai, V. J. (1997). A New Robust Model Selection Criterion for Linear Models: RFPE, unpublished note.

Yohai, V., Stahel, W. A., and Zamar, R. H. (1991). A Procedure for Robust Estimation and Inference in Linear Regression. In W. A. Stahel & S. W. Weisberg (Eds.) *Directions in Robust Statistics and Diagnostics, Part II.* New York: Springer-Verlag.

Yohai, V. J., and Zamar, R. H. (1998). Optimal Locally Robust M-estimates of Regression. *Journal of Statistical Planning and Inference*, 64, 309-323.