

**MathSoft**

---

**S-PLUS**

Documentation Supplement

Version 4.5

April 1998

Data Analysis Products Division

MathSoft, Inc.

Seattle, Washington

---

## **Proprietary Notice**

MathSoft, Inc. owns both this software program and its documentation. Both the program and documentation are copyrighted with all rights reserved by MathSoft.

The correct bibliographical reference for this document is as follows:

S-PLUS User's Guide, Data Analysis Products Division, MathSoft, Seattle, WA.

## **Copyright Notice**

Copyright ©1996-1998 MathSoft, Inc. All Rights Reserved.

Printed in the United States.

## **Acknowledgments**

S-PLUS would not exist without the pioneering research of the Bell Labs S team at AT&T (now Lucent Technologies): Richard A. Becker, John M. Chambers, Allan R. Wilks, William S. Cleveland, and colleagues.

This release of S-PLUS includes specific work from a number of scientists:

The cluster library was written by Mia Hubert, Peter Rousseeuw and Anja Struyf (University of Antwerp).

Updates to functions provided to this and earlier releases of S-PLUS were provided by Brian Ripley (Oxford University) and Terry Therneau (Mayo Clinic, Rochester).

# CONTENTS

<b>Chapter 1 Welcome to S-PLUS</b>	<b>9</b>
Introduction	9
Installation	10
System Requirements	11
<b>Help, Support, and Learning Resources</b>	<b>12</b>
Getting Help	12
<b>What's New in S-PLUS 4.5</b>	<b>16</b>
New Features	16
<b>Chapter 2 New Interactive Graphics Capabilities for S-PLUS 4.5</b>	<b>19</b>
<b>Using the Graph Tools Palette</b>	<b>20</b>
<b>Highlighting selected data points</b>	<b>24</b>
<b>Excluding or Including Only Selected Points in Your Plot</b>	<b>25</b>
<b>Color Scale Legends</b>	<b>26</b>
Usage	26
Properties	27
<b>Chapter 3 S-PLUS Excel Add-In</b>	<b>33</b>
<b>Installing the S-PLUS Excel Add-In</b>	<b>34</b>
Installation during S-PLUS setup	34
Manual installation	34
<b>Removing the S-PLUS Excel Add-in</b>	<b>37</b>
<b>Using the S-PLUS Excel Add-In</b>	<b>38</b>
<b>Selecting data for S-PLUS graphs</b>	<b>40</b>
<b>Chapter 4 S-PLUS SPSS Add-In</b>	<b>43</b>
<b>Installing the S-PLUS SPSS Add-In</b>	<b>44</b>
Installation during S-PLUS setup	44
Manual installation	44
<b>Removing the S-PLUS SPSS Add-in</b>	<b>45</b>
<b>Using the S-PLUS SPSS Add-In</b>	<b>46</b>
Selecting data for S-PLUS graphs	47
Selecting data for conditioning S-PLUS graphs	49
Handling errors during graph creation	50

<b>Chapter 5 File Improvements</b>	<b>51</b>
<b>New Input/Output Features</b>	<b>52</b>
<b>Loading Libraries</b>	<b>53</b>
<b>Loading Modules</b>	<b>54</b>
<b>Chapter 6 Manipulating Data</b>	<b>57</b>
<b>Select Data</b>	<b>58</b>
<b>Factorial Design</b>	<b>60</b>
<b>Orthogonal Array Design</b>	<b>62</b>
<b>Recode</b>	<b>64</b>
<b>Split Data By Group</b>	<b>65</b>
<b>Stack Columns</b>	<b>67</b>
<b>Subset</b>	<b>69</b>
<b>Transform</b>	<b>71</b>
<b>Transpose</b>	<b>73</b>
<b>Set Dimensions</b>	<b>74</b>
<b>Chapter 7 Resampling Methods</b>	<b>75</b>
<b>Bootstrap Inference</b>	<b>76</b>
Model Page	76
Options Page	78
Results Page	79
Plot Page	80
Jackknife-After-Bootstrap (Jack After Boot) Page	81
<b>Jackknife Inference</b>	<b>83</b>
Model Page	83
Options Page	84
Results Page	85
Plot Page	86
<b>Chapter 8 Clustering In S-PLUS</b>	<b>87</b>
<b>K-Means Clustering</b>	<b>88</b>
Model Page	88
Results Page	89
<b>Partitioning Around Medoids</b>	<b>90</b>
Model Page	90
Results Page	92
Plot Page	93

---

<b>Fuzzy Partitioning</b>	<b>94</b>
Model Page	94
Results Page	96
Plot Page	97
<b>Agglomerative Hierarchical Clustering</b>	<b>98</b>
Model Page	98
Results Page	100
Plot Page	101
<b>Divisive Hierarchical Clustering</b>	<b>102</b>
Model Page	102
Results Page	104
Plot Page	105
<b>Monothetic Clustering</b>	<b>106</b>
Model Page	106
Results Page	107
Plot Page	107
<b>Compute Dissimilarities</b>	<b>109</b>
<b>Chapter 9 Creating HTML Output</b>	<b>111</b>
<b>Tables</b>	<b>112</b>
<b>Text</b>	<b>113</b>
<b>Graphs</b>	<b>114</b>
<b>Chapter 10 Type III Sum of Squares and Adjusted Means</b>	<b>115</b>
ANOVA Tables	116
Adjusted Means	117
Multiple Comparisons	118
Estimable Functions	120
Sigma Constrained Parameterization	122
References	126
<b>Chapter 11 Power and Sample Size</b>	<b>127</b>
<b>Normal Power And Sample Size</b>	<b>128</b>
Model Page	129
Options Page	130
<b>Binomial Power And Sample Size</b>	<b>134</b>
Model Page	135
Options Page	135
Printout Page	137

<b>Power and Sample Size Theory</b>	<b>139</b>
<b>Normally Distributed Data</b>	<b>140</b>
One-Sample Test of Gaussian Mean	140
Comparing Means From Two Samples	143
<b>Binomial Data</b>	<b>146</b>
References	152
 <b>Chapter 12 Robust Linear Regression</b>	 <b>153</b>
<b>OVERVIEW OF THE ROBUST REGRESSION METHOD</b>	<b>155</b>
Key Robustness Features of the Method	155
The Essence of the Method: a Special M-Estimate	155
Using the <code>lmRobMM</code> Function to Obtain a Robust Fit	156
Comparison of Least Squares and Robust Fits	157
Robust Model Selection	157
<b>COMPUTING LEAST SQUARES AND ROBUST FITS</b>	<b>158</b>
Computing a Least Squares Fit	158
Computing a Robust Fit	159
Least Squares vs. Robust Fitted Model Objects	160
<b>VISUALIZING AND SUMMARIZING THE ROBUST FIT</b>	<b>161</b>
Visualizing the Fit with the <code>plot</code> Function	161
Statistical Inference with the <code>summary</code> Function	163
<b>COMPARING LEAST SQUARES AND ROBUST FITS</b>	<b>166</b>
Creating a Comparison Object for LS and Robust Fits	166
Visualizing LS vs. Robust Fits	166
Statistical Inference for LS vs. Robust Fits	168
<b>ROBUST MODEL SELECTION</b>	<b>170</b>
Robust F and Wald Tests	170
Robust FPE Criterion	171
<b>CONTROLLING OPTIONS FOR ROBUST REGRESSION</b>	<b>173</b>
Efficiency at Gaussian Model	173
Alternative Loss Function	173
Confidence Level of Bias Test	175
Resampling Algorithms	177
Random Resampling Parameters	177
Genetic Algorithm Parameters	178
<b>THEORETICAL DETAILS</b>	<b>179</b>
Initial Estimate Details	179
Optimal and Bisquare Rho and Psi-Functions	180

---

The Efficient Bias Robust Estimate	181
Efficiency Control	181
Robust R-Squared	181
Robust Deviance	183
Robust F Test	183
Robust Wald Test	183
Robust FPE (RFPE)	183
Appendix	184
<b>ROBUST MM REGRESSION</b>	<b>186</b>
<b>BIBLIOGRAPHY</b>	<b>198</b>
<b>Chapter 13 Parametric Regression For Censored Data</b>	<b>199</b>
<b>Introduction</b>	<b>200</b>
<b>The Generalized Kaplan-Meier Estimate</b>	<b>202</b>
Specifying Interval Censored Data	202
Computing Kaplan-Meier Estimates	204
<b>censorReg</b>	<b>207</b>
An Example Model	207
Specifying the Parametric Family	208
Accounting for Covariates	210
Truncation Distributions	212
Threshold Parameter	214
Offsets	215
Fixing parameters	216
<b>Fitting Models: ANOVA</b>	<b>218</b>
<b>Fitting Models: The plot method for CensorReg</b>	<b>220</b>
<b>Computing Probabilities and Quantiles</b>	<b>225</b>
<b>Parametric Survival</b>	<b>227</b>
Model Page	228
Options Page	230
Results Page	231
Plots Page	233
Predict Page	235
<b>Chapter 14 New GUI Toolkit Functions</b>	<b>237</b>
guiSetOption	237
guiGetOption	237
guiPrintClass	239
guiPlot	240

---

<b>Chapter 15 Automation Improvements in S-PLUS 4.5</b>	<b>245</b>
<b>Passing Data to Functions via Automation</b>	<b>246</b>
Method to get and set parameter classes of functions	247
<b>New Automation Methods in S-PLUS 4.5</b>	<b>250</b>
<b>Automating Embedded S-PLUS Graphs</b>	<b>258</b>
<b>Examples Of Automation Provided With S-plus</b>	<b>259</b>
<b>Examples Of Using S-plus As An Automation Client Included With S-plus</b>	<b>261</b>
<b>Examples of ActiveX controls included with S-PLUS</b>	<b>262</b>
 <b>Chapter 16 Dialog Controls In</b>	
<b>S-PLUS 4.5</b>	<b>263</b>
<b>ActiveX Controls in S-PLUS dialogs</b>	<b>264</b>
Adding an ActiveX control to a dialog	264
Where can the PROGID for the control be found?	265
Registering an ActiveX control	267
Why only “OCX String”?	268
Common error conditions when using ActiveX controls in S-PLUS	268
Designing ActiveX controls that support S-PLUS	269
<b>New Dialog Controls In S-PLUS 4.5</b>	<b>283</b>
 <b>Chapter 17 New Script Window Features</b>	<b>289</b>
Automatic Matching of Delimiters	289
Automatic Generation Of Right Braces	289
Automatic Indentation	290
Modifying Script Window Settings	290
 <b>Index</b>	<b>293</b>



# WELCOME TO S-PLUS

# 1

---

Introduction	9
Installation	10
System Requirements	11
<b>Help, Support, and Learning Resources</b>	<b>12</b>
Getting Help	12
<b>What's New in S-PLUS 4.5</b>	<b>16</b>
New Features	16

## Introduction

Welcome to S-PLUS Version 4.5. With many improvements to the graphical user interface introduced in S-PLUS 4.0, and many new statistical features, S-PLUS Version 4.5 offers you unparalleled power and flexibility to create innovative, cutting edge analyses.

In S-PLUS, data can be imported from virtually any source and can be viewed and edited in the Data window. Point-and-click control over the details of your graphics makes it easy to produce stunning publication quality output. Whether your task is simple or complex, S-PLUS can lead you to more insightful analysis and new discoveries.

S-PLUS is the premier solution for exploratory data analysis and statistical data mining. At the core of S-PLUS is the "S" language developed at Lucent Technologies. It is the only language created specifically for data visualization and exploration, statistical modeling, and programming with data. S provides a rich, object oriented environment designed for interactive data discovery.

As the exclusive licensee of the S language, MathSoft has molded the S technology into the most powerful data analysis product available today. The S-PLUS object-oriented environment delivers benefits that traditional language analysis programs simply can't match. With S-PLUS every data set, function, or analysis model is treated as an object, which makes it easy to examine and visually explore data, run functions one step at a time, and visually compare models for fit.

S-PLUS gives you immediate feedback because it runs functions one at a time. With S-PLUS, you've got control over every step of your analysis. Visually

compare different models for fit, re-explore your data for outliers or other factors that might influence a result, and document every analysis function. Because S-PLUS puts you in control, you'll have complete confidence in the quality of your results.

Now, even more standard analysis functions are conveniently available through menus, toolbars and dialogs, putting powerful S-PLUS techniques at your fingertips. With point-and-click ease, you can import your data, select your statistical functions and display your results. As always, when your analysis requires a new method or approach, you can modify existing methods or develop new ones with the programming language. By tapping into the power, flexibility and extensibility of S-PLUS, you can take your analysis to a new level.

## Installation

To install the software:

1. Insert the CD-ROM into your CD-ROM drive.
2. If your operating system supports AutoPlay (e.g., Windows 95 or NT 4.0), installation will proceed automatically. If not run **setup.exe** in the root directory of the CD-ROM. Use the default settings for installation.

It is a good idea to turn off other applications, in particular virus checkers, while installing S-PLUS, because of known problems with the installation software InstallShield.

If you are running a 16-bit operating system such as Windows 3.1 or Windows for Workgroups 3.11 you will need to have version 1.30.172 or higher of the Win32s subsystem on your machine, before you can install S-PLUS.

Win32s is included in the Win32s directory on the CD-ROM and may be installed by running **setup.exe** in the **Win32s\disk1** directory. Be sure to install Win32s before installing S-PLUS.

<b>Note</b>
Installing and running S-PLUS under Win32s will require approximately 50MB of combined RAM and swap file space. If you encounter the message " <b>S_apiSyncConnect Failure</b> " several times during start-up, try increasing the swap file size to 40MB in the virtual memory settings accessed through the 386 Enhanced icon in the control panel.

**Network  
Installation**

This version of S-PLUS may not be installed on a network server. If you want to run S-PLUS on a network server, contact your sales representative for a network license.

**System  
Requirements**

- Minimum platform configuration: Pentium processor with 32MB of memory.
- Hard disk space required: 61MB (Typical installation), 128MB (Full installation). Add 5MB for Adobe Acrobat Reader and 6MB for ODBC (8MB for ODBC for Win32s).
- Microsoft Windows 95, Windows NT, or Windows 3.1x
- VGA, Super VGA, or most other Windows compatible graphics cards and monitors
- One CD-ROM drive, local or networked
- Microsoft Mouse, or other Windows compatible pointing device
- Windows compatible printers are supported

## HELP, SUPPORT, AND LEARNING RESOURCES

### Getting Help

There are a variety of ways to accelerate your progress with S-PLUS, and to build upon the work of others. This section describes the learning and support resources available to S-PLUS users.

### Online Help

S-PLUS offers an online help system to make learning and using S-PLUS easier. Under the Help menu, you will find options for Using S-PLUS (how to use the graphical user interface), Language Reference (details on each function in the S-PLUS language), Questions and Answers (some common difficulties, and proposed solutions), Online Manuals (see below), and Visual Demonstrations.

There is also context-sensitive help, accessed by clicking on the Help buttons in the various dialogs, or by clicking on the context-sensitive Help button on the toolbars.

There is also Language Reference help available through the S-PLUS Commands window by typing `hel p()` at the S-PLUS prompt, or by pressing the F1 key while S-PLUS is active.

### Printed and Online Manuals

The S-PLUS *Programmer's Guide*, the *Guide to Statistics*, and the S-PLUS *User's Guide* are all available online as well as in print. To view a manual online, select Online Manuals from the S-PLUS Help menu and choose the desired title.

Notes on Online versions of the Guides
The Online manuals are viewed using Acrobat Reader, which can be installed as an option during the installation process. While using Acrobat Reader, it is generally useful to turn on <i>bookmarks</i> (under the View entry of the menu bar), rather than rely on the contents at the start of the guides. Bookmarks are always visible and can be expanded to include section headings, or collapsed to show just chapter titles.

### Online Demo

The S-PLUS Online Demos help users of all levels familiarize themselves with the new features of S-PLUS. Take a look at the user interface, learn more about common S-PLUS tasks, or show a colleague the various capabilities of S-PLUS.

### Guided Tours of S-PLUS

The S-PLUS *User's Guide* contains a tutorial, and many chapters have examples of using S-PLUS. These examples extend the techniques illustrated in the online demos.

- Add-On Modules** Add-on modules that offer analytical functionality beyond that of the base S-PLUS product include:
- S+DOX:** helps in designing and analyzing industrial experiments, especially fractional factorial experiments, response surface experiments, and robust design experiments.
- S+GARCH:** provides an essential suite of tools designed for univariate and multivariate GARCH modeling of financial time series data.
- S+SPATIALSTATS:** provides a comprehensive set of tools for statistical analysis of spatial data, including tools for hexagonal binning, variogram estimation and kriging, autoregressive and moving average modeling, and testing for spatial randomness.
- S+WAVELETS:** offers a visual data analysis approach to a whole range of signal-processing techniques, such as wavelet packets, local cosine analysis, and matching pursuits.
- StatLib** StatLib is a system for distributing statistical software, data sets, and information by electronic mail, FTP and the World Wide Web. It contains a wealth of user-contributed S-PLUS functions.
- To access StatLib by FTP, open a connection to: **lib.stat.cmu.edu**. Login as **anonymous** and send your e-mail address as your password. The FAQ (frequently asked questions) is in **/S/FAQ**, or in HTML format at **http://www.stat.math.ethz.ch/S-FAQ**.
  - To access StatLib with a web browser, visit **http://lib.stat.cmu.edu/**.
  - To access StatLib by e-mail, send the message: **send index from S to statlib@lib.stat.cmu.edu**. You can then request any item in StatLib with the request **send item from S** where **item** is the name of the item.
- S-News** S-news is an electronic mailing list by which S-PLUS users can ask questions and share information with other users. To get on this list, send a message with message body **subscribe** to **s-news-request@wubios.wustl.edu**. To get off this list, send a message with body **unsubscribe** to the same address.
- Once enrolled on the list, you will begin to receive e-mail. To send a message to the S-news mailing list, send it to: **s-news@wubios.wustl.edu**. Do *not* send subscription requests to the full list; use the s-news-request address shown above.
- Training Courses** MathSoft Educational Services offers a variety of courses designed to quickly make you efficient and effective at analyzing data with S-PLUS. The courses

are taught by professional statisticians and leaders in statistical fields. Courses feature a hands-on approach to learning, dividing class time between lecture and online exercises. All participants receive the educational materials used in the course, including lecture notes, supplementary materials, and exercise data on diskette.

## S-Press

S-Press is a free quarterly newsletter about S-PLUS mailed to primary users of S-PLUS. S-Press features stories by S-PLUS users in industry and academia, a technical support column and provides new product announcements and other information from MathSoft.

## Technical Support

In North America, to contact technical support, call  
**(206) 283-8802 ext. 235**

or fax to

**(206) 283-6310**

or send e-mail to

**support@statsci.com.**

In Europe, Asia, Australia, Africa and South America, call

**+44 1276 452299**

or fax to

**+44 1276 451224**

or email to

**shelp@mathsoft.co.uk**

## Books on Data Analysis Using S-PLUS

### General

Becker, R. A., Chambers, J. M., and Wilks, A. R. (1988). *The New S Language*. Wadsworth & Brooks/Cole, Pacific Grove, CA.

Spector, P. (1994). *An Introduction to S and S-PLUS*. Duxbury Press, Belmont, CA.

### Data Analysis

Bruce, A. and Gao, H.-Y. (1996). *Applied Wavelet Analysis with S-PLUS*. Springer-Verlag, New York.

Chambers, J. M., and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole, Pacific Grove, CA.

Everitt, B. (1994). *A Handbook of Statistical Analyses Using S-PLUS*. Chapman & Hall, London.

Härdle, W. (1991). *Smoothing Techniques with Implementation in S*. Springer-Verlag, New York.

Kaluzny, S. P., Vega, S. C., Cardoso, T. P., and Shelly, A. A. (1997). *S+SPATIALSTATS User's Manual*. Springer-Verlag, New York.

Marazzi, A. (1992). *Algorithms, Routines and S Functions for Robust Statistics*. Wadsworth & Brooks/Cole, Pacific Grove, CA.

Venables, W. N., and Ripley, B. D. (1994). *Modern Applied Statistics with S-PLUS*. Springer-Verlag, New York.

### **Graphical Techniques**

Chambers, J. M., Cleveland, W. S., Kleiner, B., and Tukey, P. A. (1983). *Graphical Techniques for Data Analysis*. Duxbury Press, Belmont, CA.

Cleveland, W. S. (1993). *Visualizing Data*. Hobart Press, Summit, NJ.

Cleveland, W. S. (1985). *The Elements of Graphing Data*. Hobart Press, Summit, NJ.

## WHAT'S NEW IN S-PLUS 4.5

The following is a summary of new features in S-PLUS 4.5. Users of S-PLUS 3.3 for Windows can browse the rest of this *User's Guide* to further acquaint themselves with the graphical user interface.

### New Features

New features and techniques include:

#### Efficiency Improvements

- Faster start-up. The Object Browser is no longer started by default (although you can still obtain this behavior by modifying the settings in the Startup page of the General Settings dialog).
- Faster data entry. A new option, Buffer Data Entry, on the General page of the General Settings dialog under the Options menu, allows S-PLUS to buffer changes in a Data Sheet, so that they are sent to the data engine in chunks, rather than as they are made, as in S-PLUS 4.0. Set this option to Off to restore the old behavior.
- Menu item and function to refresh memory.

#### Statistics

- New robust regression method available via menu system and through the `lmRobMM` function.
- Power and sample size calculations.
- Enhanced parametric survival (accelerated failure time) estimation.
- Type III sums of squares for ANOVA.
- Bootstrap and jackknife estimation now available through the menu system.
- Clustering methods are now available through the menu system.



## Graphics

- Interactively select and highlight points.
- Redraw excluding selected points.
- Linked highlighting in scatter plots.
- Interactively rescale axes (pan,crop).
- Trellis drill-down: select one panel of a Trellis graph and create a full-size copy.
- Color scale legends.

## User Interface

- Select Data dialog at startup for easy data selection.
- More data manipulation dialogs:
  - Recode
  - Split
  - Stack
  - Subset
  - Transform
  - Transpose
  - Factorial Design
  - Orthogonal Array Design
  - Set Dimensions
- Improved Insert Graph dialog with enhanced thumbnails.
- Excel add-in for S-PLUS graphics.

## Import and Export

- New Excel add-in to create S-PLUS graphics from Excel.

- Function to generate HTML tables.

### **Programming**

- Dialogs now support ActiveX controls.
- Enhanced automation support, including many new included examples.
- Enhanced editing features in script windows include automatic delimiter matching and auto-indent feature.

### **License Management**

- License manager for network version.

# NEW INTERACTIVE GRAPHICS CAPABILITIES FOR S-PLUS 4.5


# 2

---

<b>Using the Graph Tools Palette</b>	<b>20</b>
<b>Highlighting selected data points</b>	<b>24</b>
<b>Excluding or Including Only Selected Points in Your Plot</b>	<b>25</b>
<b>Color Scale Legends</b>	<b>26</b>
Usage	26
Properties	27

## USING THE GRAPH TOOLS PALETTE

A new Graph Tools Palette has been introduced in S-PLUS 4.5. It provides tools for selecting data, refocusing the graph on a subregion or specific panel, and for creating and modifying interactive Trellis graphs. Some of these tools were also available on the Annotations or the Plot2D palettes in S-PLUS 4.0.

To bring up the graph tools palette, click on the Graph Tools button  on the graph sheet toolbar (shown when a graph sheet is in focus).

To enable any of the tools, click on the appropriate button on the Graph Tools Palette.



Select Tool

Standard selection mode, where clicking on a graphical object selects that object.



Label Point

Click on any point in a 2D scatter plot to label it with its row name.



Select Data

Click on any point or drag a rectangle around a group of points in a scatter plot to select them. They will appear selected in the scatter plot, in any other scatter plots using the same data set, and in any grid views of the data set. Points can be added to the selection by pressing the CTRL key when releasing the mouse button. This tool can also be accessed from the annotation tools palette. The way in which the data points are highlighted in the scatter plot can be defined on the Interactive page of the Graphs dialog. To open this dialog, choose Graph Options from the Options menu.



Crop Graph to Selected Rectangle

Drag a rectangle around the area of a 2D graph on which you would like to refocus. The X and Y axes will be rescaled to show only this area of the graph. This tool can also be accessed from the Rescale Axes menu option in the graph sheet Format menu.



**Auto Scale Axes** Clicking on this button will reset the axes scaling for both the X and the Y axes to include all points (i.e., Auto minimums and maximums are used). This can also be done by selecting Reset Auto Scaling menu from the graph sheet Format menu.



**Pan Up**

If you have cropped your graph to show only a subsample, use this button to show the region directly above the current region. The amount of overlap between regions can be specified on the Interactive page of the Graphs dialog. To open this dialog, choose Graph Options from the Options menu.



**Pan Right**

If you have cropped your graph to show only a subsample, use this button to show the region directly to the right of the current region.



**Pan Left**

If you have cropped your graph to show only a subsample, use this button to show the region directly to the left of the current region.



**Pan Down**

If you have cropped your graph to show only a subsample, use this button to show the region directly below the current region.



**Extract Panel**

Use this button if you would like to extract a single panel from a conditioned graph. After clicking on the tool button, click anywhere within the panel that you would like to extract. Conditioning for the graph will be turned off, and the “Subset Rows with” expression for the plots will be set to the conditioning expression for that panel. This can also be done by choosing Extract Panel/Redraw Graph from the graph sheet Format menu. To have the panel placed in a separate graph sheet, select Extract Panel/New Graph Sheet from the graph sheet Format menu.



**Return To All Panels**

Use this button if you have extracted a panel and would like to return to the full conditioned graph. The Panel Type for the graph will be set to Conditioned, and the Subset Rows expressions for all plots will be set to ALL. Alternatively, choose Show All Panels from the graph sheet Format menu.



#### No Conditioning

Use this button to set the Panel Type for a graph to None. All plots will be within a single plot area.



#### 4 Panel Conditioning

Use this button to set the Panel Type for a graph to Conditioned, and to set the number of panels used for continuous data to 4. If no conditioning data has been specified, no panels will be drawn.



#### 9 Panel Conditioning

Use this button to set the Panel Type for a graph to Conditioned, and to set the number of panels used for continuous data to 9. If no conditioning data has been specified, no panels will be drawn.



#### Plots in Separate Panels

If you are plotting more than one set of data series on a graph, use this button to have each plot drawn in a separate panel. The Panel Type for the graph will be set to By Plot. The axes scaling for the X and Y axes will be the same for each panel.



#### Separate Panels with Varying Y Axes

If you are plotting more than one set of data series on a graph, use this button to have each plot drawn in a separate panel. The Panel Type for the graph will be set to By Plot. The axes scaling for the X axis will be the same for each panel, but the Y axis ranges will vary according to the data in each panel. The Number of Columns for the panels is set to Auto, which defaults to 1 so that the panels will appear one above the other.



### Separate Panels with Varying X Axes

If you are plotting more than one set of data series on a graph, use this button to have each plot drawn in a separate panel. The Panel Type for the graph will be set to By Plot. The axes scaling for the Y axis will be the same for each panel, but the X axis ranges will vary according to the data in each panel. The Number of Rows for the panels is set to Auto, which defaults to 1 so that the panels will appear side by side.



### Panels with Varying X and Y Axes

If you are plotting more than one set of data series on a graph, use this button to have each plot drawn in a separate panel. The Panel Type for the graph will be set to By Plot. The axes scaling for the X and Y axes will vary according to the data in each panel.

# HIGHLIGHTING SELECTED DATA POINTS

You can modify the way in which points are highlighted in the Interactive page of the Graphs dialog. To open this dialog, choose Graph Options from the Options menu. The options are:

**Display Selected Points**

If this box is not checked, scatter plot points will not be highlighted. The remainder of the fields in this dialog will not be used.

The options for specifying the selected symbols are:

- |                       |   |
|-----------------------|---|
| Style                 | Choose a symbol style for the selected symbols. If None is chosen, the selected symbols will be the same style as is specified for the plot.  |
| Color                 | Choose a color for the selected symbols. If Transparent is chosen, the same symbol color will be used as is specified for the plot.   |
| Height Multiplier     | Choose a multiple for increasing the size of the symbol. If 1.0 is specified, the selected symbols will be the same size as is specified for the plot.  |
| Line Weight Increment | Choose the amount to increase the line weight used in drawing the symbol above what is specified for the plot. If Hairline is chosen, the line weight of the selected symbols will be the same as the plot. |



# EXCLUDING OR INCLUDING ONLY SELECTED POINTS IN YOUR PLOT

Three new menu options are available under the Format menu when a graph sheet is in focus.

## Exclude Selected Points

Selecting this menu option will remove any currently selected points from your plots. If your plots require calculations, such as for smoothing, the calculations will be redone excluding the selected points. An expression defining the currently selected rows is put into the Subset Rows with field of the Data to Plot page of the plot dialog. Because the plot is excluding the selected points, the expression begins with a minus sign. Any previous subsetting specifications will be replaced. Further changes in data point selections will not alter the plot.

## Use Only Selected Points

Selecting this menu option will remove all points from your plot except those that are selected. If your plots require calculations, such as for smoothing, the calculations will be redone with only the selected points. An expression defining the currently selected rows is put into the Subset Rows with field on the Data to Plot page of the plot dialog. Any previous subsetting specifications will be replaced. Further changes in data point selections will not alter the plot.

## Include All Points

Selecting this menu options will turn off subsetting for the plots on the graph sheet. The Subset Rows field on the Data to Plot page of the plot dialog will be set to "ALL". Any previous subsetting specifications will be replaced.

## COLOR SCALE LEGENDS

### Usage

The Color Scale Legend may be used with any of the following GUI plot types:

- Line Plot
- Area Plot
- Bar Plot
- Contour Plot
- Surface Plot
- QQ Plot
- 3D Line Plot
- Pie Plot
- Scatter Plot Matrix

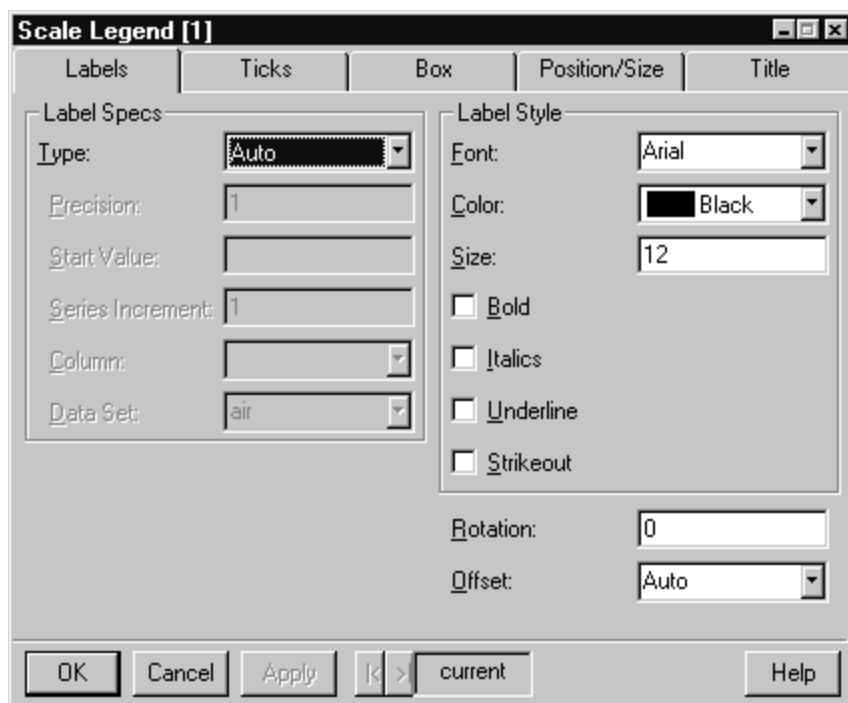
The Color Scale Legend button located on Graph toolbar activates when a plot of an appropriate plot type is selected. Selection is implicit if the required plot is the only plot in a GraphSheet and nothing is selected. Otherwise the plot must be selected specifically.

## Properties

When you choose the Color Scale Legend button from the Graph toolbar, S-PLUS automatically creates a color scale legend. To control the color scale legend, use the Scale Legend dialog, which you can access by double-clicking on the color scale legend.

## Labels Page

The Labels page allows the users to specify the output format and font for the labels on the scale bar. Rotation and offset of the labels from the bar are specified here as well. All of these options work the same as the labels for 2D axes.



**Ticks Page**      The Ticks page allows users to specify the properties of the ticks on the scale bar. The options work the same as with a 2D axis.

Scale Legend [1]

Labels

Ticks

Box

Position/Size

Title

Scale Range

Minimum: Auto

Maximum: Auto

Interval

Interval: Auto

Interval Type: Auto

Column:

Data Set: air

Tick Range

First Tick: Auto

Last Tick: Auto

Ticks

Length: 0.04

Weight: 1/4

OK

Cancel

Apply

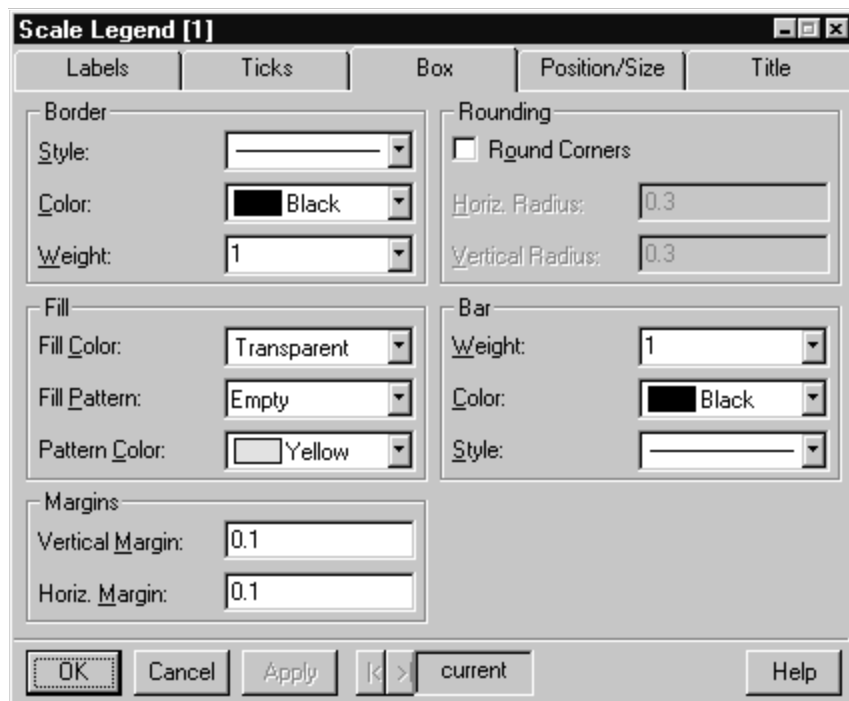
< >

current

Help

**Box Page**

The Box page gives control over the format of the outer border of the scale legend as well as the scale bar.



The image shows a dialog box titled "Scale Legend [1]" with five tabs: Labels, Ticks, Box, Position/Size, and Title. The "Box" tab is selected. The dialog is divided into several sections for configuring the legend's appearance.

**Border**

- Style:** A dropdown menu showing a solid line.
- Color:** A color selection button showing a black square, with the text "Black" next to it.
- Weight:** A dropdown menu showing the value "1".

**Fill**

- Fill Color:** A dropdown menu showing "Transparent".
- Fill Pattern:** A dropdown menu showing "Empty".
- Pattern Color:** A color selection button showing a yellow square, with the text "Yellow" next to it.

**Margins**

- Vertical Margin:** A text input field containing "0.1".
- Horiz. Margin:** A text input field containing "0.1".

**Rounding**

- Round Corners:** An unchecked checkbox.
- Horiz. Radius:** A text input field containing "0.3".
- Vertical Radius:** A text input field containing "0.3".

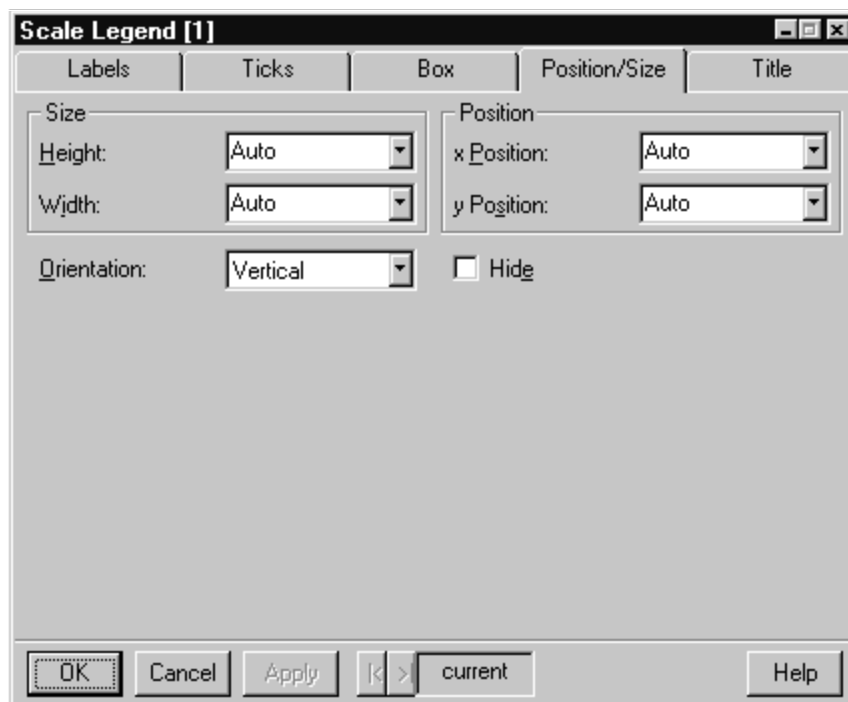
**Bar**

- Weight:** A dropdown menu showing "1".
- Color:** A color selection button showing a black square, with the text "Black" next to it.
- Style:** A dropdown menu showing a solid line.

**Buttons:** At the bottom, there are buttons for "OK", "Cancel", "Apply", navigation arrows "<" and ">", a "current" button, and a "Help" button.

## Position/Size Page

The Position/Size page allows users to specify the position and size of the scale legend as well as the orientation. The default is to position the legend on the right side of the corresponding graph when using a vertical orientation. For a horizontal orientation the default is to position the legend across the top of the graph.



One option on the Position/Size page deserves special mention. The Hide checkbox allows you to format and store a color scale legend without rendering it to the screen or in print. You may, for example, be creating a graph that will be used in different situations, in some of which you want a legend, and in some of which you don't.

### To hide a legend

1. Check the Hide checkbox and click OK.

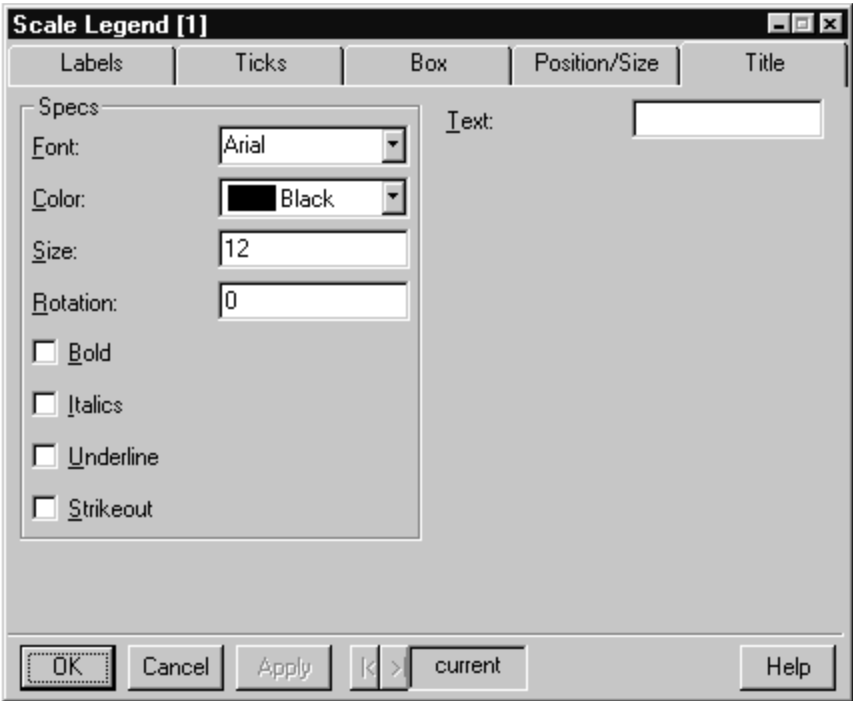
### To show a hidden legend

1. Use the Object Browser to find the plot containing the hidden legend.
2. Click on the plot in the left pane to view the objects within the plot

listed in the right pane.

3. Click on Scale Legend in the right pane to bring up the Color Scale Legend properties dialog.
4. Click the Position/Size tab.
5. Uncheck the Hide checkbox and click OK.

**Title Page**      The Title page allows you to specify and format a title for the legend. The default is to not display a title.





---

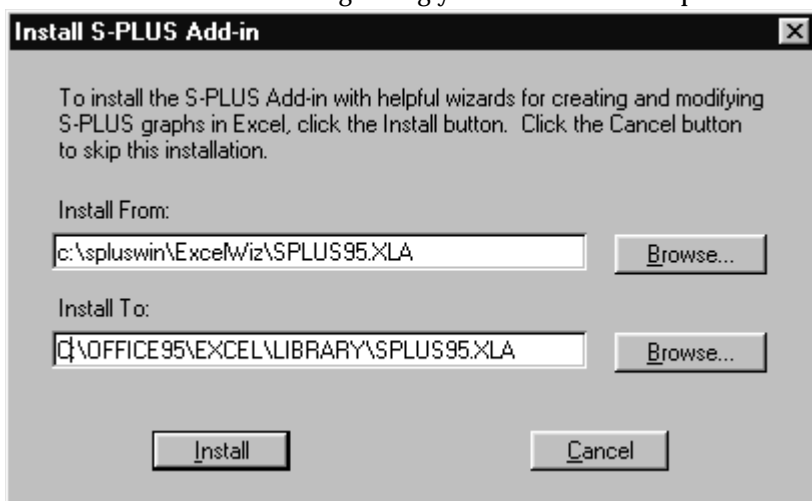
<b>Installing the S-PLUS Excel Add-In</b>	<b>34</b>
Installation during S-PLUS setup	34
Manual installation	34
<b>Removing the S-PLUS Excel Add-in</b>	<b>37</b>
<b>Using the S-PLUS Excel Add-In</b>	<b>38</b>
<b>Selecting data for S-PLUS graphs</b>	<b>40</b>

New to S-PLUS 4.5 is a Microsoft Excel add-in application that makes it easier to create and modify S-PLUS graphs from within Microsoft Excel. This add-in includes the ability to create S-PLUS graphs from selected data, to modify the layout of an S-PLUS graph embedded in Excel, and to modify the properties of a plot in an embedded S-PLUS graph in Excel. A helpful wizard guides you through the process of selecting data, choosing an S-PLUS graph and plot type and creating the graph in Excel, much like Excel's ChartWizard.

## INSTALLING THE S-PLUS EXCEL ADD-IN

### Installation during S-PLUS setup

During a typical, custom, or server installation of S-PLUS 4.5, S-PLUS setup will examine your system for an appropriate version of Microsoft Excel. The S-PLUS Excel Add-in requires Microsoft Excel version 7.0 or higher. Once detected, setup will automatically enable the option to install this add-in. You can disable installation of the add-in by choosing the custom install and un-checking this option from the list of options. At the end of setup, you will be prompted to install the S-PLUS Add-in. Setup will then start Excel and load a special add-in installation program in Excel to continue with installation. You will see a dialog asking you to confirm some paths:



You can use the browse buttons to change the paths detected. Click the “Install” button to install the S-PLUS Add-in in Excel. When completed, you will see a successful completion dialog. Close Excel to continue with the rest of S-PLUS setup.

### Manual installation

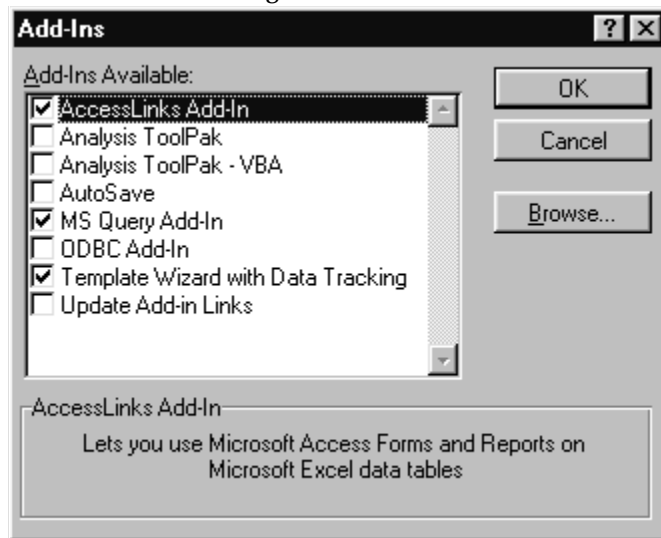
If you choose not to install the S-PLUS Excel Add-in during S-PLUS setup, you can install this option at any later time using S-PLUS setup and choosing the custom setup mode. Then, select the S-PLUS Add-in from the list of custom setup options.

If you installed the S-PLUS Excel Add-in on a server, you can install the add-in on a workstation without using S-PLUS setup. Open the file called **INSTALL.XLA** from the **ExcelWiz** subdirectory of the S-PLUS program

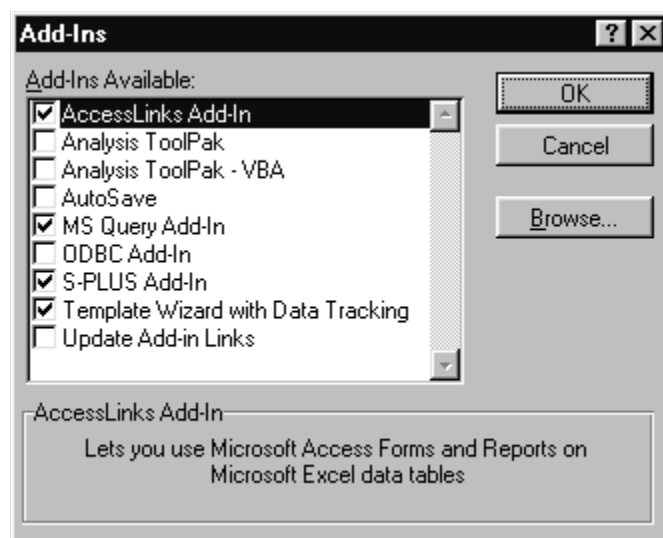
directory on the server system. This will start an automatic installation of the add-in from the server to the workstation.

You can also manually install the add-in using Excel. To do this follow these steps:

1. Start Microsoft Excel 7.0 or higher on your workstation.
2. Create a new worksheet if one does not already exist.
3. From the Tools menu select “Add Ins...”
4. From the Add Ins dialog click the browse button:



5. For Excel 7.0, select “SPLUS95.XLA” on the server in the “ExcelWiz” subdirectory of the S-PLUS program directory. For Excel 8.0 or higher, select “SPLUS97.XLA”.
6. You may be prompted to copy the S-PLUS Add-in application from its location on your server system to the Excel library directory on your workstation. You may choose to copy or not.
7. A check box next to the name of the S-PLUS Add-in will now appear in the Add Ins dialog list of add-ins. You may now click the OK button to dismiss this dialog.



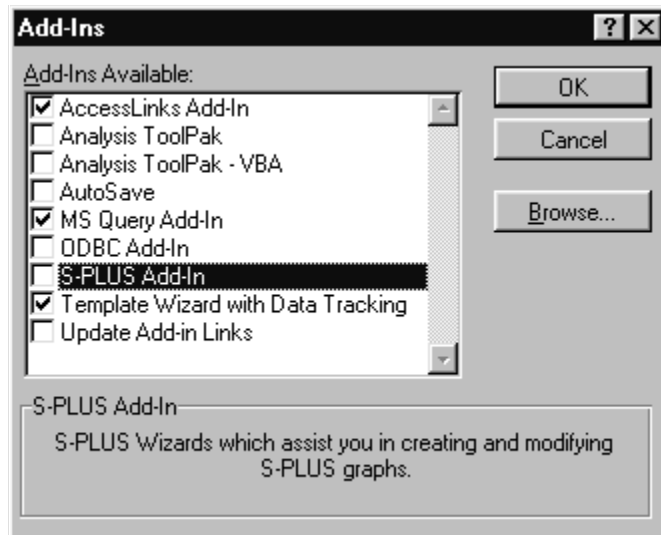
## REMOVING THE S-PLUS EXCEL ADD-IN

If you installed the S-PLUS Excel Add-in during S-PLUS setup, when you choose to remove S-PLUS, this add-in will automatically be removed from Excel by S-PLUS setup.

If you manually installed the add-in, such as in the case of a workstation as detailed above, you will need to manually remove this add-in from Excel. Open the file called **REMOVE.XLA** from the **ExcelWiz** subdirectory of the S-PLUS program directory on the server system. This will start an automatic removal of the add-in from your workstation.

You can also remove the add-in using Excel. To do this follow these steps:

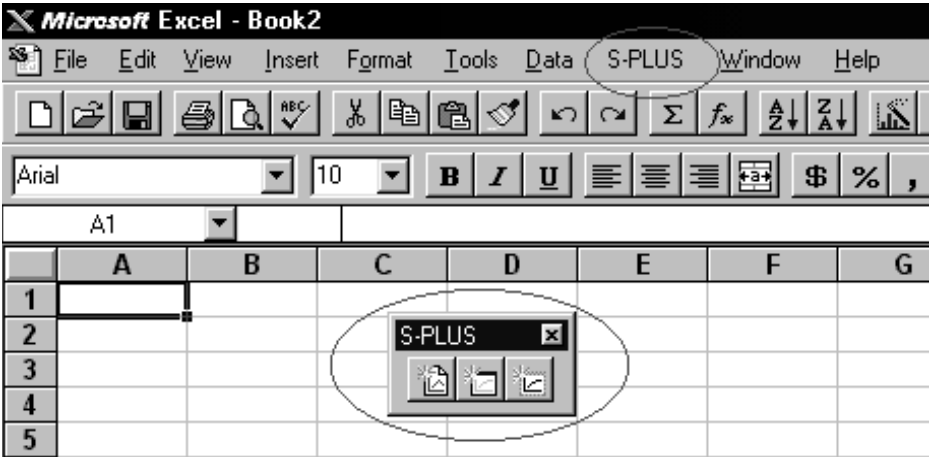
1. Start Microsoft Excel 7.0 or higher.
2. Create a new worksheet if one does not already exist.
3. From the Tools menu select “Add Ins...”
4. From the Add Ins dialog select the “S-PLUS Add-in” option in the list of add-ins and un-check this option:



5. The add-in will be unloaded from Excel. The add-in file may still remain in your Excel library directory on disk. You may have to manually delete this file.

# USING THE S-PLUS EXCEL ADD-IN

When installed in Excel, whenever you have a worksheet in focus, the following menu and toolbar will be available:



There are several options on the toolbar and in the menu:

**Table 1:**

To create a new S-PLUS graph with the currently selected data in the current worksheet:



1. Select blocks of data in the current worksheet you want to graph with S-PLUS.
2. Click on this button or select the “Create Graph” option from the S-PLUS menu.
3. Follow the wizard to create the graph.

**Table 1:****To modify the layout properties of the currently selected S-PLUS graph:**

1. Select an S-PLUS graph in your worksheet by clicking once on it. (If you double-click on an S-PLUS graph you will activate it and start editing in place)
2. Click on this button or select the “Modify Graph Layout” option in the S-PLUS menu.
3. An S-PLUS graph sheet layout dialog will appear in Excel allowing you to modify any of the layout properties of this graph.

**To modify the properties of a plot in the currently selected S-PLUS graph:**

1. Select an S-PLUS graph in your worksheet by clicking once on it.
2. Click on this button or select the “Modify Plots” option in the S-PLUS menu.
3. A dialog will appear showing you a list of the graph areas in this graph (you can have multiple graph areas in a graph, i.e. one graph area might be 2D and another might be 3D in the same graph) and for each graph area, a list showing all the plots in this graph area.
4. Select the graph area and the plot in this area you want to edit. Click next.
5. An S-PLUS plot properties dialog will appear in Excel allowing you to modify properties of this plot.

## SELECTING DATA FOR S-PLUS GRAPHS

Before you can create a graph, you must first select data in your current worksheet. You must select a block of data that is greater than one cell in width or length before you can continue with the Create Graph wizard. S-PLUS plots accept data in a variety of formats. Some S-PLUS plots require at least three columns of data and the data are interpreted as X, Y, and Z data values. Other plots require at least four columns of data and interpret the data selected as X, Y, Z, and W data values. The list of plot types in the last page of the Create Graph wizard indicates what kind of data specification is required. If no X, Y, Z, or W specification is shown for a plot type in the list, that means it accepts X single or multiple columns or X and Y data with single or multiple columns. For an explanation of the data specifications for various S-PLUS plot types, please see the *S-PLUS User's Guide*, Chapter 8 *Creating a Graph, Preparing Data for Graphing*.

### Warning

You should typically not select an entire column in Excel as part of a data spec for an S-PLUS graph, because Excel will send all rows of this column to S-PLUS for graphing, whether the rows are empty or not. This may cause errors in S-PLUS or a failure to create the graph.

The S-PLUS Excel Add-in fully supports multiple column and row selections and discontinuous block selections in Excel to specify data for an S-PLUS graph. For example, if you wanted to create two line plots in an S-PLUS graph and you had the following data in Excel:

	A	B	C	D	E	
1	1	2	3	5	1	
2	2	4	1	4	2	
3	3	2	3	3	1	
4	4	4	1	2	2	
5	5	2	3	2	1	
6	6	4	1	1	2	
7						

you could select the data:



	A	B	C	D	E
1	1	2	3	5	1
2	2	4	1	4	2
3	3	2	3	3	1
4	4	4	1	2	2
5	5	2	3	2	1
6	6	4	1	1	2
7					

The Create Graph wizard will treat the A column in this selection as the X data, and the B and C columns as the Y data. This will create two line plots, the first one with X data as the A column and Y data as the B column, next second plot with X data as the A column and Y data as the C column.

You could also have selected the same data using discontinuous column selection:

	A	B	C	D	E
1	1	2	3	5	1
2	2	4	1	4	2
3	3	2	3	3	1
4	4	4	1	2	2
5	5	2	3	2	1
6	6	4	1	1	2
7					

In this case rows 1 to 6 in column A were first selected, the control key was held down, then the block from B1 to C6 was selected. This selection will produce the same graph with two plots as the above example.

If a plot expects only one column of data for a given dimension, such as the X data for a line plot, and more than one column is included in the selection, only the first column in the selection will be sent to S-PLUS to make the graph. For example, using the above data, you select the blocks A1:B6 and C1:D6:

	A	B	C	D	E
1	1	2	3	5	1
2	2	4	1	4	2
3	3	2	3	3	1
4	4	4	1	2	2
5	5	2	3	2	1
6	6	4	1	1	2
7					

The Create Graph wizard will send A1:A6 as the X data, C1:D6 as the Y data to create two line plots.

### ***Selecting data for conditioning S-PLUS graphs***

When you are using the Create Graph wizard, in step 2 you can specify an Excel worksheet and data range to use for conditioning the graph you are creating. A conditioned graph allows you to view your data in a series of panels, where each panel contains a subset of the original data. The subset in each panel is determined by the levels of the conditioning data range you select. You can skip conditioning by leaving the “Conditioning range” edit field in this dialog blank.

When specifying a data range for conditioning, you may specify any valid data range in normal Excel range syntax. For example, say you specified the data range A1:B6 from the Sheet1 worksheet for the data to plot in a S-PLUS graph. You could also specify the data range C1:C6 from Sheet1 for the conditioning data. If a 2D line plot is created, the plot will be conditioned on the data in C1:C6.

### ***Handling errors during graph creation***

If S-PLUS encounters problems during the creation of a graph in Excel, any error messages will appear in a modeless dialog box in Excel. If errors occur, it might mean that invalid data was specified for the plot created. It might also indicate another problem related to the range or data type of the data specified. The graph may not be created if errors occur. Please see the *S-PLUS User's Guide* for explanations of error messages.

---

<b>Installing the S-PLUS SPSS Add-In</b>	<b>44</b>
Installation during S-PLUS setup	44
Manual installation	44
<b>Removing the S-PLUS SPSS Add-in</b>	<b>45</b>
<b>Using the S-PLUS SPSS Add-In</b>	<b>46</b>
Selecting data for S-PLUS graphs	47
Selecting data for conditioning S-PLUS graphs	49
Handling errors during graph creation	50

New to S-PLUS 4.5 is an add-in application that works with SPSS to make it easier to create and modify S-PLUS graphs from within SPSS. This add-in includes the ability to create S-PLUS graphs from selected variables in the SPSS data editor, to modify the layout of an S-PLUS graph embedded in a SPSS output document, and to modify the properties of a plot in an embedded S-PLUS graph in SPSS. A helpful wizard guides you through the process of selecting variables, choosing an S-PLUS graph and plot type and creating the graph in SPSS.

## INSTALLING THE S-PLUS SPSS ADD-IN

### **Installation during S-PLUS setup**

During typical, custom, or server installation of S-PLUS 4.5, S-PLUS setup will examine your system for an appropriate version of SPSS. The S-PLUS SPSS Add-in requires SPSS version 8.0 or higher. Once detected, setup will automatically enable the option to install this add-in. You can disable installation of the add-in by choosing the custom install and un-checking this option from the list of options. At the end of setup, you will be prompted to install the S-PLUS Add-in. Setup will then start a special installation program for this add-in. Just follow the steps in the installation program. When completed, you will see a successful completion dialog.

### **Manual installation**

If you choose not to install the S-PLUS SPSS Add-in during S-PLUS setup, you can install this option at any later time using S-PLUS setup and choosing the custom setup mode. Then, select the S-PLUS Add-in for SPSS from the list of custom setup options.

If you installed the S-PLUS SPSS Add-in on a server, you can install the add-in on a workstation without using S-PLUS setup. Run the file called "Setup.exe" from the "SPSSWiz" subdirectory of the S-PLUS program directory on the server system and follow the steps to install the add-in.

---

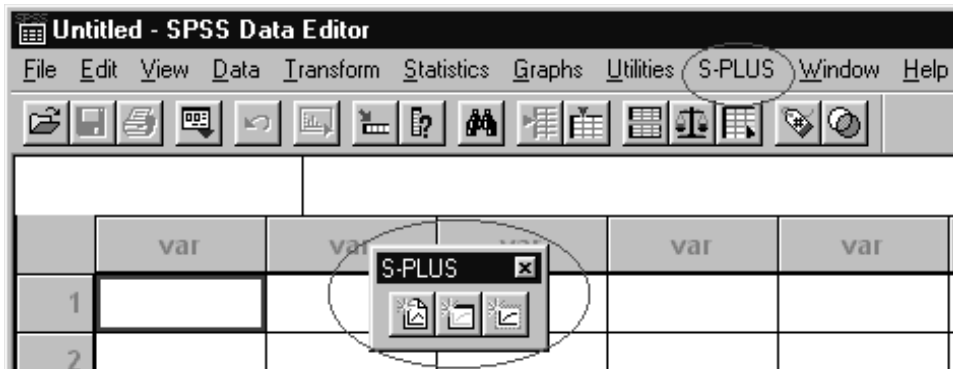
## REMOVING THE S-PLUS SPSS ADD-IN

If you installed the S-PLUS SPSS Add-in during S-PLUS setup, when you choose to remove S-PLUS, this add-in will automatically be removed from SPSS by S-PLUS setup.

If you manually installed the add-in, such as in the case of a workstation as detailed above, you will need to manually remove this add-in from SPSS. Run the file called "Setup.exe" from the "SPSSWiz" subdirectory of the S-PLUS program directory on the server system and follow the steps to remove the add-in.

## USING THE S-PLUS SPSS ADD-IN

When installed in SPSS, whenever you have the data editor open, the following menu and toolbar will be available:



The same menu and toolbar are also available whenever you have an output document open. S-PLUS graphs created with this add-in are placed in an output document. You have a choice to create a new output document or to use an existing one.

There are several options on the toolbar and in the menu:



**Create a new S-PLUS graph with the currently selected variables in the data editor.**

To use this option follow these steps:

1. Select variables in the data editor you want to graph with S-PLUS.
2. Click on this button or select the "Create Graph" option from the S-PLUS menu.
3. Follow the wizard to create the graph.



### **Modify the layout properties of the currently selected S-PLUS graph.**

To use this option follow these steps:

1. Select an S-PLUS graph in an output document by clicking once on it. (If you double-click on an S-PLUS graph, you will activate it and start editing in place.)
2. Click on this button or select the “Modify Graph Layout” option in the S-PLUS menu.
3. An S-PLUS graph sheet layout dialog will appear in SPSS allowing you to modify any of the layout properties of this graph.



### **Modify the properties of a plot in the currently selected S-PLUS graph.**

To use this option follow these steps:

1. Select an S-PLUS graph in an output document by clicking once on it.
2. Click on this button or select the “Modify Plots” option in the S-PLUS menu.
3. A dialog will appear showing you a list of the graph areas in this graph (you can have multiple graph areas in a graph, i.e., one graph area might be 2D and another might be 3D in the same graph) and for each graph area, a list showing all the plots in this graph area.
4. Select the graph area and the plot in this area you want to edit. Click next.
5. An S-PLUS plot properties dialog will appear in SPSS allowing you to modify properties of this plot.

## **Selecting data for S-PLUS graphs**

Before you can create a graph, you must first select data in the data editor. You can select variables in the SPSS data editor by clicking on the column header where the variable name appears for each variable you want to include in the graph. S-PLUS plots accept data in a variety of formats. Some S-PLUS plots require at least three columns of data and the data are interpreted as X, Y, and Z data values. Other plots require at least four columns of data and interpret the data selected as X, Y, Z, and W data values. For an explanation of the data specifications for various S-PLUS plot types, please see the *S-PLUS User's Guide*, Chapter 8 *Creating a Graph, Preparing Data for Graphing*.

For example, if you wanted to create two line plots in an S-PLUS graph and you had the following variables in SPSS:

	xdata	ydata1	ydata2	ydata3	
1	1.00	2.00	5.00	5.00	
2	2.00	4.00	3.00	2.00	
3	3.00	2.00	2.00	2.00	
4	4.00	4.00	3.00	3.00	
5	5.00	2.00	4.00	1.00	
6	6.00	4.00	2.00	.00	
7					

you could select the variables 'xdata', 'ydata1', and 'ydata2' for graphing:

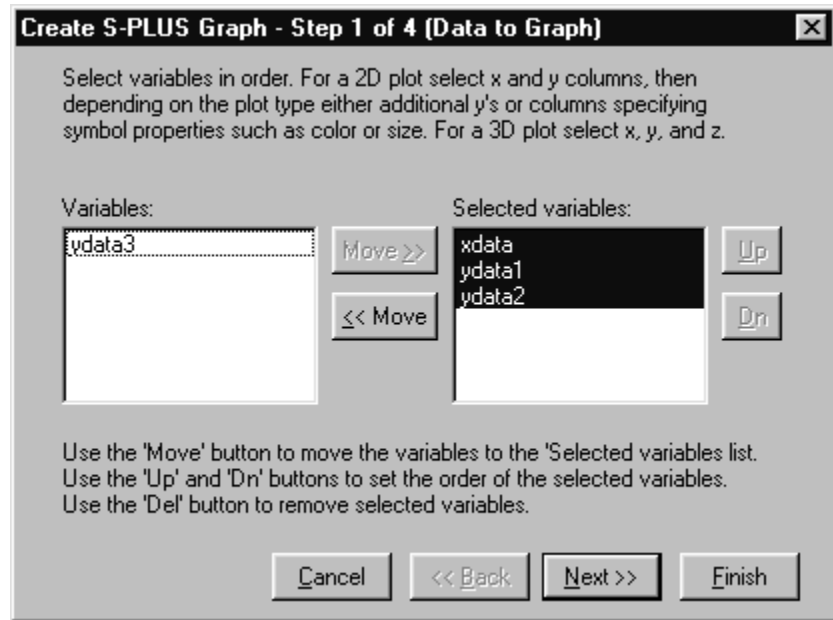
	xdata	ydata1	ydata2	ydata3	
1	1.00	2.00	5.00	5.00	
2	2.00	4.00	3.00	2.00	
3	3.00	2.00	2.00	2.00	
4	4.00	4.00	3.00	3.00	
5	5.00	2.00	4.00	1.00	
6	6.00	4.00	2.00	.00	

The Create Graph wizard will treat the 'xdata' variable in this selection as the X data and the 'ydata1' and 'ydata2' variables as the Y data. This will create two line plots, the first one with X data as the 'xdata' variable and Y data as the 'ydata1', the second plot with X data as the 'xdata' variable and Y data as the 'ydata2' variable.

Steps 1 and 2 of the Create Graph wizard allow you to add to, remove from,



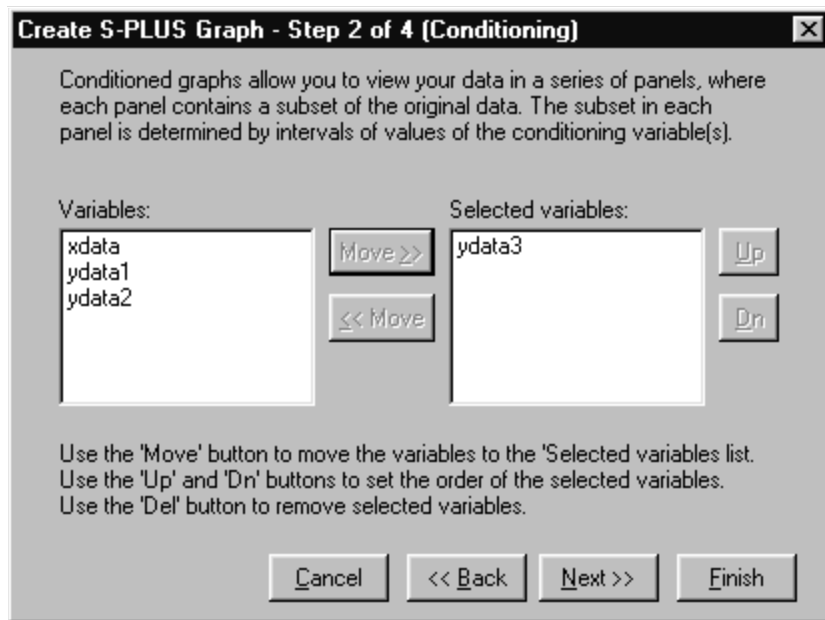
and re-order the list of selected variables to use to create an S-PLUS graph:



The list called 'Variables' on the right is the list of all available variables in the data editor. The list called 'Selected variables' is a list of variables from the available variables you've chosen to include in the S-PLUS graph. When a variable name is selected in either list, you can use the Move buttons to move it between the lists. When a variable name is selected in the 'Selected variables' list, you can use the Up and Dn buttons to change the order of the variables. The order of the selected variables is important because the order will determine how S-PLUS graphs the data. A similar dialog allows you to select variables for conditioning the graph you create.

### Selecting data for conditioning S-PLUS graphs

When you are using the Create Graph wizard, in step 2 you can specify variables to use for conditioning the graph you are creating:



A conditioned graph allows you to view your data in a series of panels, where each panel contains a subset of the original data. The subset in each panel is determined by the levels of the conditioning data range you select. You can skip conditioning by leaving the 'Selected variables' list in this dialog empty.

## Handling errors during graph creation

If S-PLUS encounters problems during the creation of a graph in SPSS, any error messages will appear in a modeless dialog box in SPSS. If errors occur, it might mean that invalid data were specified for the plot created. It might also indicate another problem related to the range or data type of the data specified. The graph may not be created if errors occur. Please see the S-PLUS User's Guide for explanations of error messages.

# FILE IMPROVEMENTS

# 5

---

<b>New Input/Output Features</b>	<b>52</b>
<b>Loading Libraries</b>	<b>53</b>
<b>Loading Modules</b>	<b>54</b>

S-PLUS Version 4.5 includes several small improvements to make file operations smoother and easier. These include improved support for spreadsheet and database import, and menu options for loading modules and libraries.

## NEW INPUT/OUTPUT FEATURES

Importing data using the Import Data dialog has been enhanced for S-PLUS 4.5 by the following new features:

- A separate entry for Foxpro files has been added to the Files of type field. Some users experienced difficulty using the Dbase/Foxpro option on S-PLUS 4.0.
- A Page field has been added to the Options page for file types that support paged data (e.g., Excel). This allows you to specify from which page of a multi-page spreadsheet you wish to read. Previously, S-PLUS always read from the first page.
- A Name Col field has been added to the Options page. This field allows you to designate one column of the imported data to be used as rownames in the same way that the Name Row field allows you to designate a row for use as column names.
- An Import Text as Factors field has been added to the Options page. By default, text columns shorter than 250 rows are read as factors; longer columns are read as character data. If you set this field to “Never”, all text columns are imported as character data. If you set this field to “Always”, all text columns are imported as factor data.

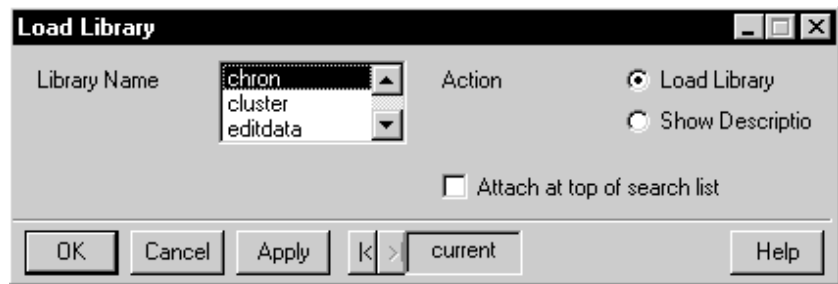
Editing data in the Data Window has been speeded up by the use of a special buffer; changes in the Data Window are not transmitted to the S-PLUS engine immediately, but rather are recorded in the buffer and transmitted to the engine in chunks of approximately 50 edits. An option in the Options menu's General Settings dialog, Buffer Data Entry, controls whether this special buffer is used. If you turn off Buffer Data Entry, editing commands are transmitted immediately as in S-PLUS 4.0.

# LOADING LIBRARIES

S-PLUS includes a number of function *libraries* that extend basic functionality or provide instructive examples of S-PLUS programming. Some of these libraries, such as the cluster library and the GUI library, are loaded (or *attached*) automatically when you start S-PLUS. You can attach other libraries as needed using the Load Library dialog.

## To load an S-PLUS library

1. From the File menu, select Load Library. The Load Library dialog appears as shown below:



2. From the Library Name scrolled list, select the library you want to load.
3. If you want the library functions to appear before other system files in the search list, select the checkbox labeled “Attach at top of search list.”
4. Select the Load Library radio button.
5. Click OK.

## To view a brief description of a library's contents

1. From the File menu, select Load Library. The Load Library dialog appears.
2. From the Library Name scrolled list, select the library you want to load.
3. Select the Show Description radio button.
4. Click OK. Notepad will appear with the library's description.

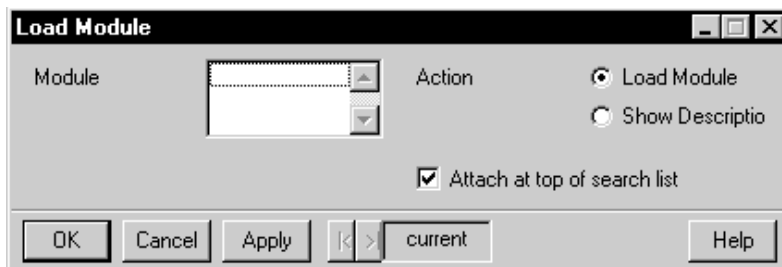
## LOADING MODULES

MathSoft offers a number of add-on *modules* for S-PLUS that provide comprehensive solutions in specific subject areas. Currently, the following modules are available:

- S+SPATIALSTATS for the exploration and modeling of spatially correlated data.
- S+WAVELETS for wavelet analysis of signals, time series, images, and other data.
- S+GARCH for modeling financial and econometric data.
- S+DOX for design and analysis of experiments.

### To load an S-PLUS module

1. From the File menu, select Load Module. The Load Module dialog appears as shown below:



2. From the Module scrolled list, select the module you want to load.
3. If you want the module functions to appear before other system files in the search list, select the checkbox labeled “Attach at top of search list.” For modules, this is the recommended (and default) behavior because some modules redefine system functions to ensure correct behavior.
4. Select the Load Module radio button.
5. Click OK.

**To view a brief description of a module's contents**

1. From the File menu, select Load Module. The Load Module dialog appears.
2. From the Module scrolled list, select the module you want to load.
3. Select the Show Description radio button.
4. Click OK. Notepad will appear with the module's description.





# MANIPULATING DATA

# 6

---

<b>Select Data</b>	<b>58</b>
<b>Factorial Design</b>	<b>60</b>
<b>Orthogonal Array Design</b>	<b>62</b>
<b>Recode</b>	<b>64</b>
<b>Split Data By Group</b>	<b>65</b>
<b>Stack Columns</b>	<b>67</b>
<b>Subset</b>	<b>69</b>
<b>Transform</b>	<b>71</b>
<b>Transpose</b>	<b>73</b>
<b>Set Dimensions</b>	<b>74</b>

## SELECT DATA

This dialog provides a convenient mechanism for selecting data for use in analyses.

To select data:

Choose **Data:Select Data** from the main menu. The dialog shown below appears.

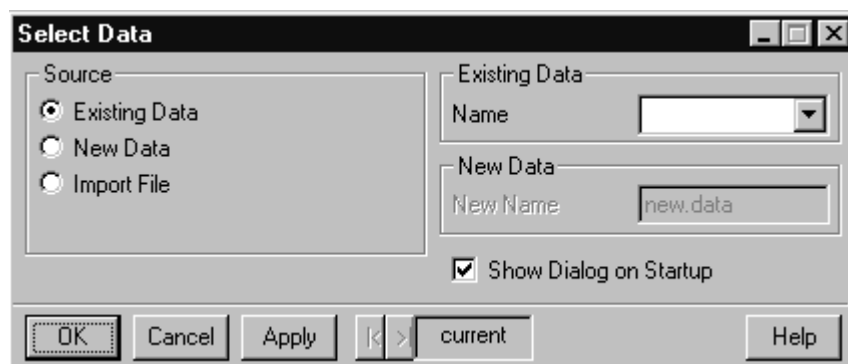


Figure 6.1: *The Select Data dialog.*

**Source** Select “Existing Data” to view an existing data frame in a Data Window. Select “New Data” to create a new data frame and display it in a Data Window. Select “Import File” to launch the Import Data dialog.

**Existing Data** **Name**  
Specify the name of the existing data frame to display. Note that the drop-down list will contain all user-created data sets which are in the S-PLUS working database. Built-in example data such as **fuel.frame** and **environmental** do not appear in the list, but may be specified by typing in the name.

**New Data** **New Name**  
Name for new data frame. If the name of an existing data frame is specified, the existing data frame will be displayed.

---

### Show Dialog on Startup

Check box indicating whether to display this dialog whenever S-PLUS is started. This option may also be specified on the **Startup** page of the **General Settings** options dialog. You can also use the **Startup** page to specify whether you want an Object Browser and/or Commands window to display on startup.

## FACTORIAL DESIGN

This dialog creates a factorial or fractional factorial design.

To create a factorial design:

Choose **Data:Design:Factorial** from the main menu. The dialog shown below appears.

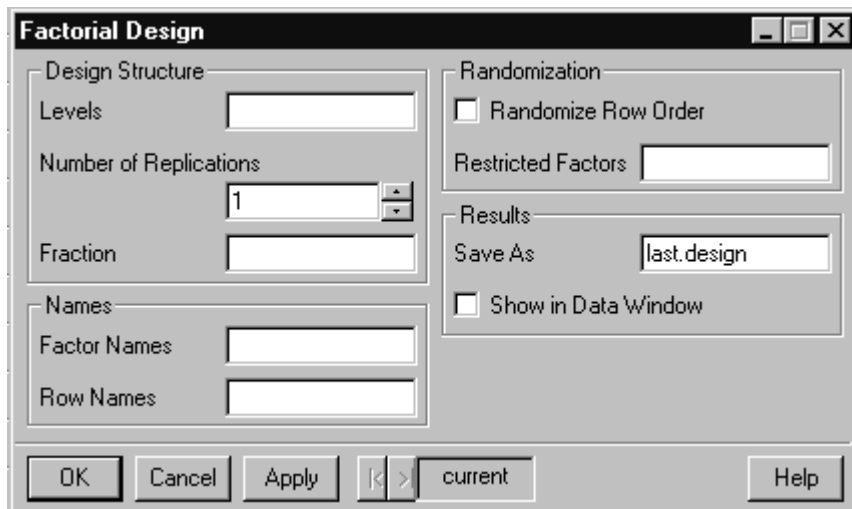


Figure 6.2: *The Factorial Design dialog.*

### Design Structure Levels

Enter a vector of the number of levels for the factors in the design. For example, to generate a design with three levels of one variable and two levels of another, specify `c(3, 2)`.

### Number of Replications

Specify the number of times the complete design should be replicated.

### Fraction

Optionally, specify the definition for the fraction desired in a fractional

factorial design. This may either be a numerical fraction (e.g.  $1/4$  for a quarter replicate), or a model formula giving one or more defining contrasts (e.g.,  $A: B: D + B: C: E$ ). Fractional factorials are provided only for 2-level factors. By default, a full factorial design is created.

## Names

### Factor Names

Optionally, specify names for the factors. This may be a vector of character strings which are the names of the factors. It may also be a list, in which case the names attribute of the list is the names of the factors, and the components of the list (which need not be of mode character) label the levels of the corresponding factor. If factor names are not given, they default to A, B, etc. If levels are not given, they default to the factor name (possibly abbreviated) followed by level numbers.

### Row Names

Optionally, specify names to use for the rows of the design. The default is  $1:nrows$ , where  $nrows$  is the number of observations in the design.

## Randomization

### Randomize Row Order

Check here to randomize the order of the rows in the design.

### Restricted Factors

Optionally, specify a vector (either numeric or character) naming some factors (columns) in the design which shouldn't be scrambled.

## Results

### Save As

Enter the name for the object in which to save the results of the analysis. If an object with this name already exists, its contents will be overwritten.

### Show in Data Window

Check this box to display the new design in a Data Window.

### Related programming language functions:

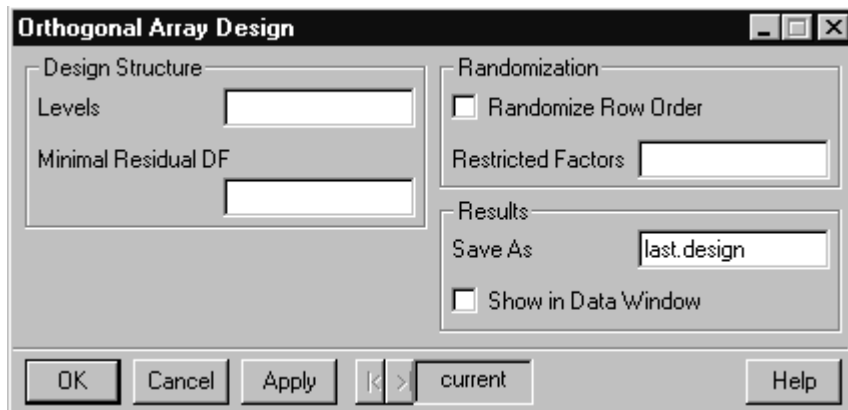
`fac.design`, `randomize`

## ORTHOGONAL ARRAY DESIGN

This dialog generates an orthogonal array design.

To generate an orthogonal array design:

Choose **Data:Design:Orthogonal Array** from the main menu. The dialog shown below appears.



**Figure 6.3:** *The Orthogonal Array Design dialog.*

### Design Structure    Levels

Enter a vector of the number of levels for the factors in the design. For example, to generate a design with three levels of one variable and two levels of another specify `c(3, 2)`.

### Minimal Residual DF

Optionally, specify the minimum residual degrees of freedom requested for a main-effects-only model. The default value is 0, unless the number of levels in the factors are all equal in which case the default is 3.

### Randomization    Randomize Row Order

Check here to randomize the order of the rows in the design.

**Restricted Factors**

Optionally, specify a vector (either numeric or character) naming some factors (columns) in the design which shouldn't be scrambled.

**Results****Save As**

Enter the name for the object in which to save the results of the analysis. If an object with this name already exists, its contents will be overwritten.

**Show in Data Window**

Check this box to display the new design in a Data Window.

**Related programming language functions:**

[oa.design](#), [randomize](#)

# RECODE

This dialog recodes all occurrences of a specific value in specified columns to a specified new value.

To recode a value:

Choose **Data:Recode** from the main menu. The dialog shown below appears.

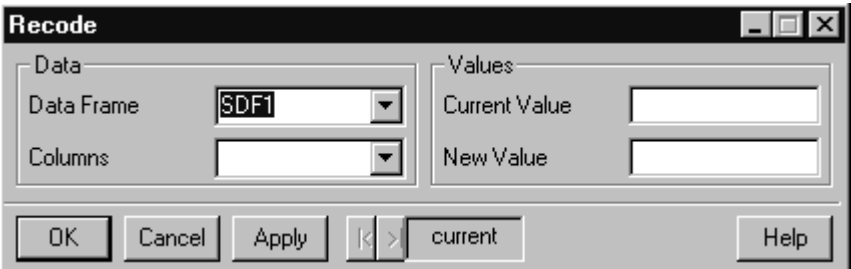


Figure 6.4: The Recode dialog.

<b>Data</b>	<b>Data Frame</b> Select or enter the name of the data frame containing the columns to be recoded.
	<b>Columns</b> Select the columns to recode.
<b>Values</b>	<b>Current Value</b> Value to be changed. If the column being recoded is a factor, then this will be coerced to a character string.
	<b>New Value</b> New value used to replace all occurrences of the <b>Current Value</b> in the specified columns.

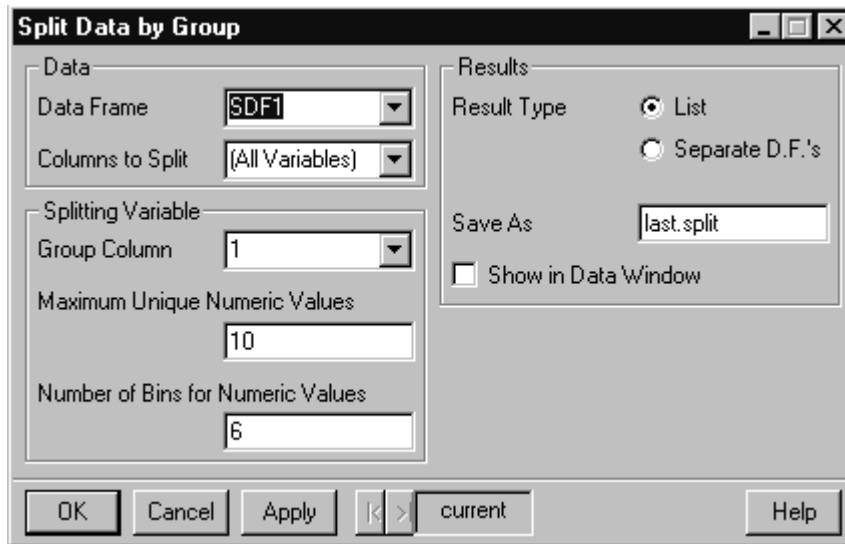


# SPLIT DATA BY GROUP

This dialog splits a data frame into multiple new data frames based on the values of a splitting variable.

To split a data frame:

Choose **Data:Split** from the main menu. The dialog shown below appears.



**Figure 6.5:** *The Split Data by Group dialog.*

## Data

### Data Frame

Specify the data frame to split into separate data frames.

### Columns to Split

Specify the columns to include in the new data frames. By default, all columns are included.

## Splitting Variable

### Group Column

Specify the column to use as the splitting column. If this column is a factor, a

new data frame will be created for each level of the factor. If this column is numeric, the number of new data frames is determined by “Maximum Unique Numeric Values” and “Number of Bins for Numeric Values”.

### Maximum Unique Numeric Values

If the **Group Column** is numeric with at most **Maximum Unique Numeric Values** unique values, then a new data frame will be created for each unique value of the column. If there are more than **Maximum Unique Numeric Values** unique values, then the data will be split into **Number of Bins for Numeric Values** new data frames by classifying the grouping variable into the specified number of bins of equal width.

### Number of Bins for Numeric Values

Number of new data frames to create when the **Group Column** is numeric and contains more than **Maximum Unique Numeric Values** unique values.

## Results

### Result Type

Select “List” to return a list containing the new data frames as components. Select “Separate D.F.’s” to return separate data frames. When “Separate D.F.’s” is selected, a warning message will be issued giving the names of the new data frames.

### Save As

Specify the name for the results. If the **Result Type** is specified to be “List”, this is the name of the list. If the **Result Type** is “Separate D.F.’s”, names are constructed for the new data frames by concatenating this name with the name of the appropriate level in the grouping variable.

### Show in Data Window

Check this box to display the new data frames in Data Windows. This is only available when the **Result Type** is “Separate D.F.’s”

**Related programming language functions:**

[split](#)

# STACK COLUMNS

This dialog stacks separate columns of a data frame into a single column, with the values of other columns replicated appropriately.

To stack data frame columns:

Choose **Data:Stack** from the main menu. The dialog shown below appears.

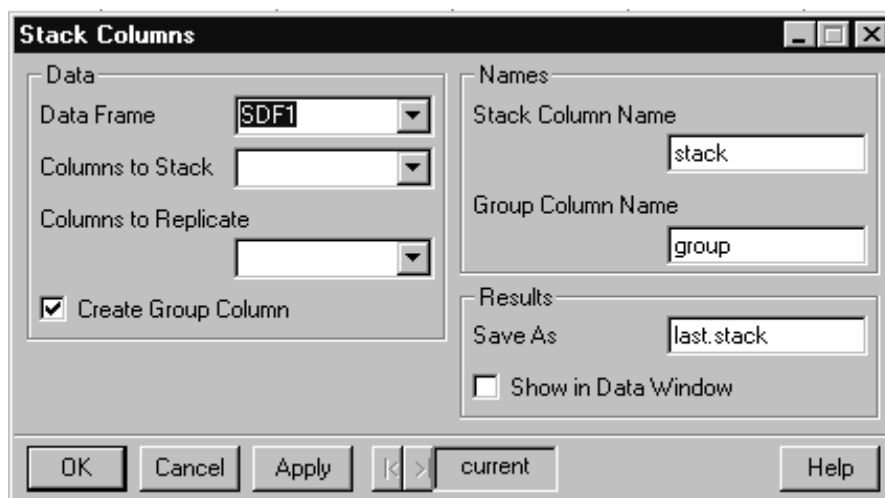


Figure 6.6: *The Stack Columns dialog.*

## Data

### Data Frame

Specify the data frame.

### Columns to Stack

Specify the column giving the variables to be stacked. A new column will be created by stacking the selected columns.

### Columns to Replicate

Specify the columns to include in the new data frames. By default, no columns are replicated.

**Create Group Column**

Check this to add a factor column giving group membership for each stacked value. The column names of the stacked columns are used as the factor levels.

**Names****Stack Column Name**

Specify the name for the new column containing the stacked data.

**Group Column Name**

Specify the name for the group membership column. This is only relevant if **Create Group Column** is checked.

**Results****Save As**

Enter the name for the data frame to contain the stacked data.

**Show in Data Window**

Check this box to display the new data frame in a Data Window.

# SUBSET

This dialog creates a subset of a data frame based on a subsetting expression. The subset may be indicated by a logical column or a new data frame containing the subset of the data may be created.

To subset a data frame:

Choose **Data:Subset** from the main menu. The dialog shown below appears.

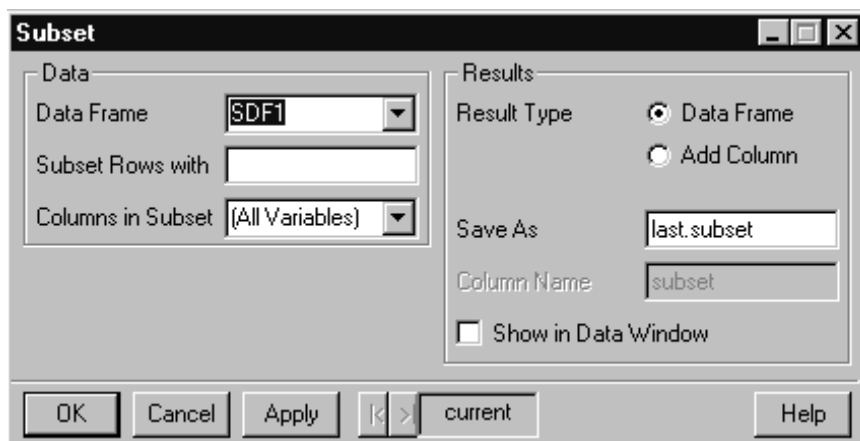


Figure 6.7: *The Subset dialog.*

## Data

### Data Frame

Specify the data frame.

### Subset Rows with

Enter an S-PLUS expression that identifies the rows to include in the subset. The expression must evaluate to a vector of logical values (TRUE values are used, FALSE values are dropped) or a vector of indices identifying the numbers of the rows to use.

**Columns in Subset**

Select the columns to be included in the new data frame. By default, all columns are included.

**Results****Result Type**

Select “Data Frame” to return a new data frame containing only the specified subset of rows. Select “Add Column” to return all rows with a new logical column indicating subset membership.

**Save As**

Enter the name for the new data frame. If an object with this name already exists, its contents will be overwritten.

**Column Name**

Specify the name for the new column indicating subset membership. This is only relevant if **Result Type** is “Add Column”.

**Show in Data Window**

Check this box to display the new data frame in a Data Window.

# TRANSFORM

This dialog creates a new variable based on a transformation of other variables.

To transform variables:

Choose **Data:Transform** from the main menu. The dialog shown below appears.

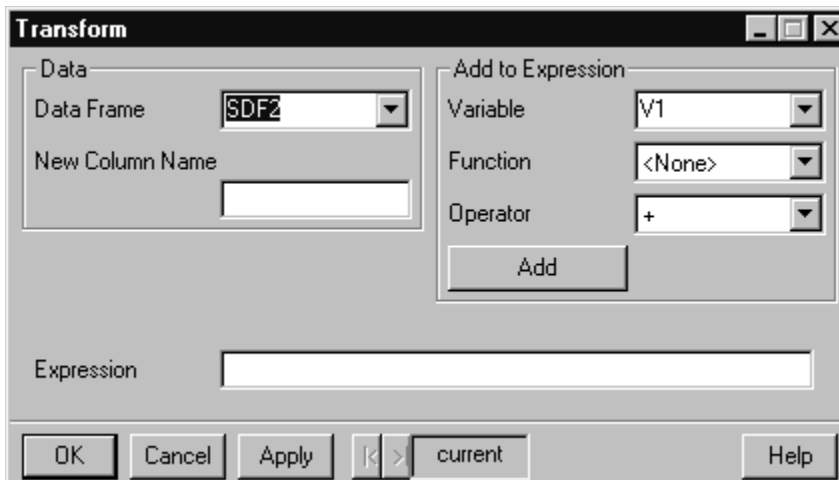


Figure 6.8: *The Transform dialog.*

<b>Data</b>	<b>Data Frame</b>
	Specify the data frame.
	<b>New Column Name</b>
	Specify a name for the new column containing the transformed data.
	<b>Expression</b>
	Specify an expression describing the transform. This expression may refer to other columns in the data frame. The expression may be typed in or built using the <b>Add to Expression</b> controls.

**Add to  
Expression****Variable**

Select a variable to use in the expression.

**Function**

Select a function to apply to the specified **Variable** in the expression. Generally this will be a function that takes a single vector as input and returns a vector or scalar. If the result is a scalar, the value will be replicated to match the length of the result vector, as is standard in S-PLUS. If the function takes more than one argument, the other arguments will be written in the expression with their default values. These values may be edited in the **Expression** field.

**Operator**

Select the operator to use when placing the new term in the expression.

**Add**

Press this to append the new term to the expression.

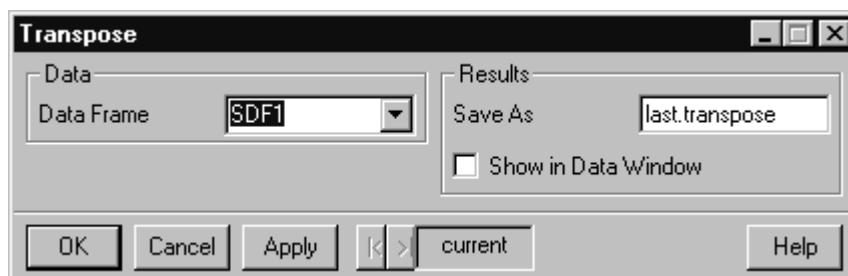


# TRANSPOSE

This dialog transposes a data frame.

To transpose a data frame:

Choose **Data:Transpose** from the main menu. The dialog shown below appears.



**Figure 6.9:** *The Transpose dialog.*

## Data

### Data Frame

Specify the data frame to transpose. Generally this will be a completely numeric data frame. Transposing a data frame containing factors will produce a data frame in which all columns are factors, with each unique value in each new column being a factor level.

## Results

### Save As

Enter the name for the new data frame. If an object with this name already exists, its contents will be overwritten.

### Show in Data Window

Check this box to display the new data frame in a Data Window.

**Related programming language functions:**

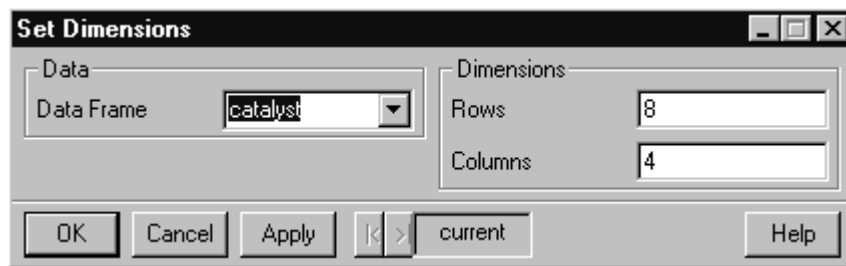
[t](#)

## SET DIMENSIONS

This dialog sets the number of rows or columns in a data frame, matrix, or vector. Empty cells will be filled with missing values (NA's). Specifying the extent of the data object before entering data will speed data entry in a Data Window.

To set the dimensions of a data object:

Choose **Data:Set Dimensions** from the main menu. The dialog shown below appears.



**Figure 6.10:** *The Set Dimensions dialog.*

<b>Data</b>	<p><b>Data Frame</b></p> <p>Specify the data frame. The dimensions of this data frame will be extended to the specified extents.</p>
<b>Dimensions</b>	<p><b>Rows</b></p> <p>Specify the desired number of rows. If this value is less than the current number of rows in the data frame, it will be ignored.</p> <p><b>Columns</b></p> <p>Specify the desired number of columns. If this value is less than the current number of columns in the data frame, it will be ignored.</p> <p><b>Related programming language functions:</b></p> <p><a href="#">di m</a></p>

---

# RESAMPLING METHODS

# 7

---

<b>Bootstrap Inference</b>	<b>76</b>
Model Page	76
Options Page	78
Results Page	79
Plot Page	80
Jackknife-After-Bootstrap (Jack After Boot) Page	81
<b>Jackknife Inference</b>	<b>83</b>
Model Page	83
Options Page	84
Results Page	85
Plot Page	86

## BOOTSTRAP INFERENCE

This dialog performs bootstrap inference for a specified statistic and data frame. See chapter 30 in the *Guide to Statistics* for details.

To perform bootstrap inference:

Choose **Statistics:Resample:Bootstrap** from the main menu. The dialog shown below will appear.

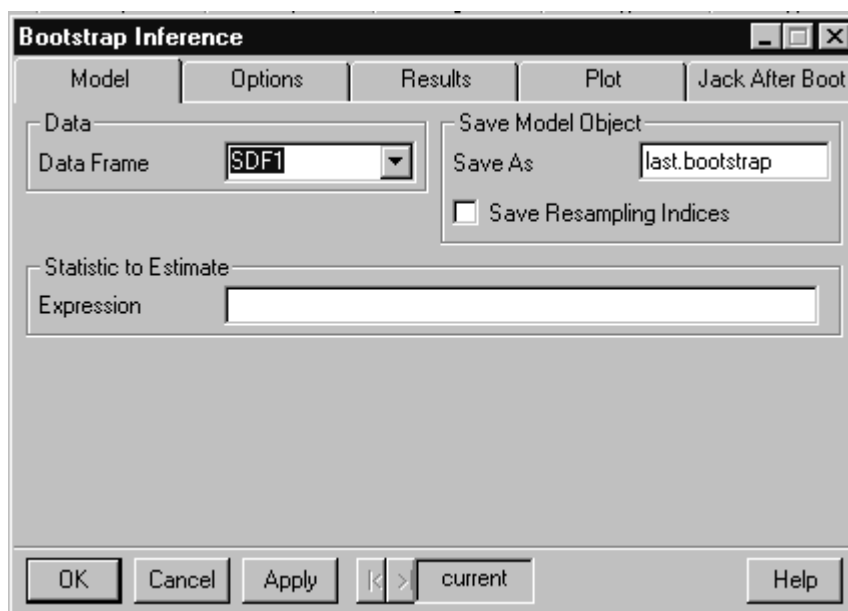


Figure 7.1: *The Bootstrap Inference dialog, Model page.*

### Model Page

#### Data

#### Data Frame

Specify the data to bootstrap. This may be a vector, matrix, or data frame.

<b>Statistic to Estimate</b>	<p><b>Expression</b></p> <p>Specify the expression describing the statistic to be bootstrapped. It may be a function that accepts data as the first argument and returns a vector or matrix, or a call referring to the data that evaluates to a vector or matrix.</p> <p>For example, to bootstrap the regression coefficients for regressing <code>Mileage</code> on <code>Weight</code> in the <code>fuel.frame</code> data, use the expression <code>coef(lm(Mileage~Weight, fuel.frame))</code> and specify <code>fuel.frame</code> as the <b>Data Frame</b>. To bootstrap the mean of <code>Mileage</code>, use the expression <code>mean(Mileage)</code>.</p>
<b>Save Model Object</b>	<p><b>Save As</b></p> <p>Enter the name for the object in which to save the results of the analysis. If an object with this name already exists, its contents will be overwritten.</p> <p><b>Save Resampling Indices</b></p> <p>Check this to save the matrix of resampling indices describing which observations appear in each resample.</p>

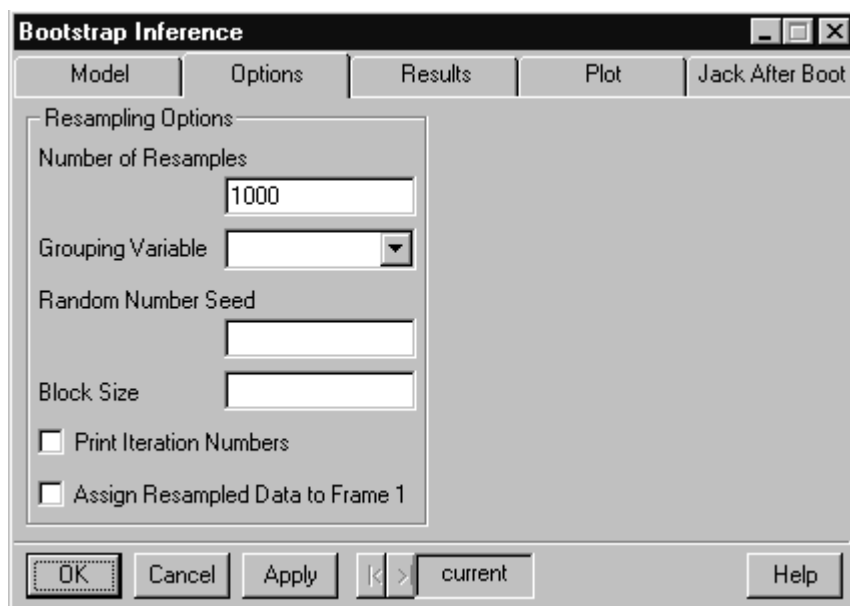


Figure 7.2: The Bootstrap Inference dialog, Options page.

## Options Page

### Resampling Options

#### Number of Resamples

Specify the number of replicates to draw. The default is 1000 replicates as this is a minimal number recommended for estimating percentiles.

#### Grouping Variable

Specify a grouping variable to use when resampling observations. If this is specified, sampling will be done within each group so that subgroup proportions in the resamples match those of the original sample. This provides inference condition upon subgroup size.

#### Random Number Seed

Specify an integer between 0 and 1000 to set the random number seed to a desired value. Specifying the seed allows a way to obtain identical results from multiple bootstrap runs.

#### Block Size

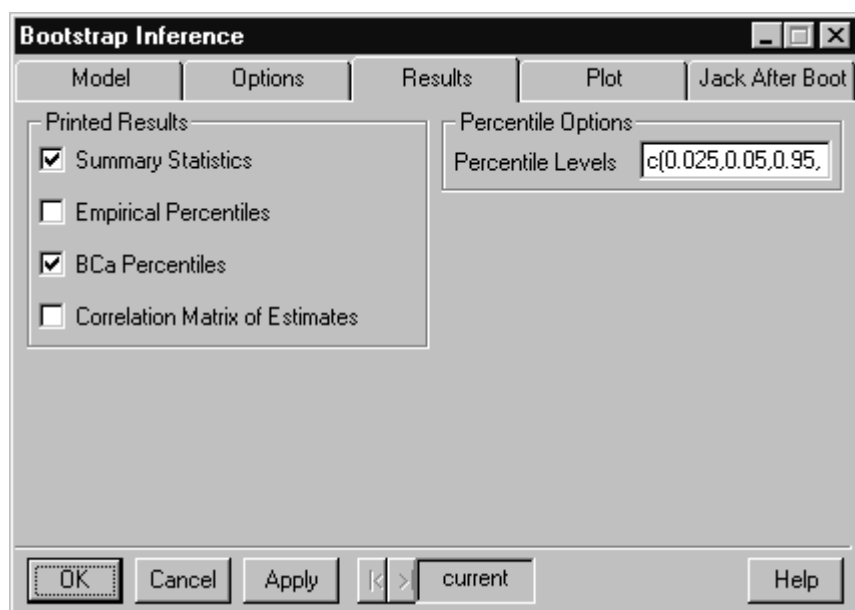
Specify the block size to use when calling the sampling function. See the language help for [bootstrap](#) for details.

#### Print Iteration Numbers

Check this to display iteration progress by printing iteration number ranges. Due to the timing of output display, this is not as useful from the dialog as from the command line function call.

#### Assign Resampled Data to Frame 1

Check this to assign the resampled data to frame 1 as each sample is generated. See the language help for [bootstrap](#) for details.



**Figure 7.3:** *The Bootstrap Inference dialog, Results page.*

## Results Page

### Printed Results

#### Summary Statistics

Check this to print basic summaries such as the bootstrap estimates of bias, mean, and standard error.

#### Empirical Percentiles

Check this to print empirical percentiles for the statistic under consideration.

#### BCa Percentiles

Check this to print BCa percentiles for the statistic under consideration. Note that BCa percentiles are generally more accurate than empirical percentiles.

**Correlation Matrix of Estimates**

Check this to print the correlation matrix for the estimates. Note that this is only relevant if the statistic under consideration is a vector, such as a vector of regression coefficients.

**Percentile Options**

**Percentile Levels**

Specify a vector of percentile levels at which to evaluate the empirical or BCa percentiles..

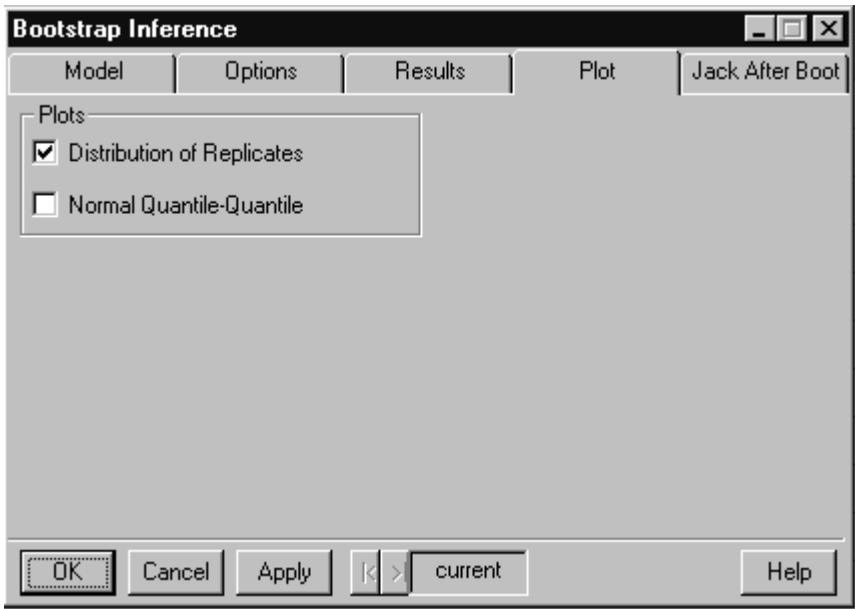


Figure 7.4: *The Bootstrap Inference dialog, Plot page.*

**Plot Page**

**Plots**

**Distribution of Replicates**

Check this to plot the distribution of the replicates for each statistic of interest.



## Normal Quantile-Quantile

Check this to plot a Normal quantile-quantile plot for each statistic of interest.

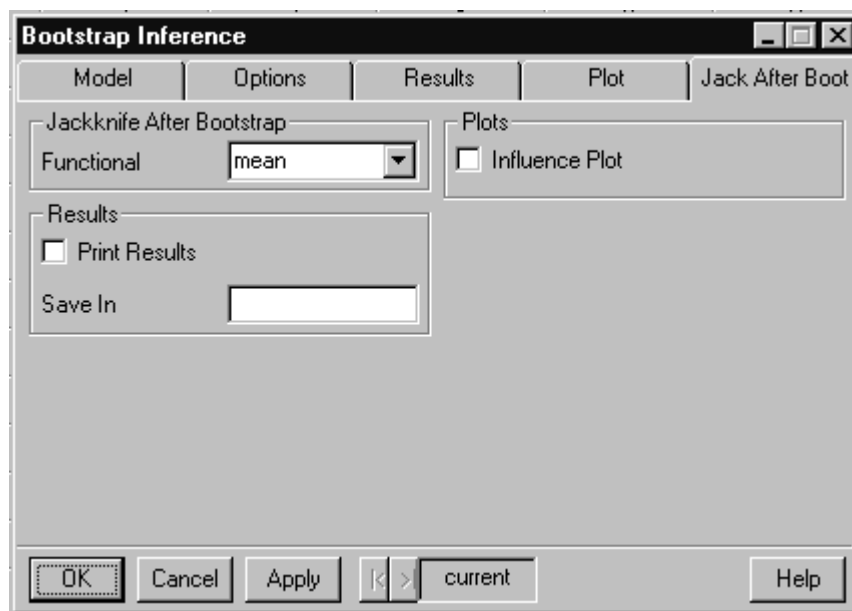


Figure 7.5: *The Bootstrap Inference dialog, Jack After Boot page.*

### Jackknife- After- Bootstrap (Jack After Boot) Page

Jackknife-after-bootstrap is a technique applied to the results of a bootstrap analysis which is used to get estimates of variability and influence for some functional of the distribution of bootstrap replicates. It is useful for determining which observations most influence the bootstrap results, and for getting estimates of standard error for bootstrap statistics.

#### Jackknife After Bootstrap

##### Functional

Specify the functional to apply to the distribution of replicates. This may be “Mean”, “Bias”, “SE”, or the name of a function such as [max](#).

## Results

### Print Results

Check this to print the jackknife-after-bootstrap summaries.

### Save In

Enter the name for the object in which to save the jackknife-after-bootstrap results. If an object with this name already exists, its contents will be overwritten.

## Plots

### Influence Plot

Check this to plot a jackknife-after-bootstrap influence plot indicating the degree of influence of each observation on the bootstrap results.

**Related programming language functions:**

[bootstrap](#), [jack.after.bootstrap](#)

# JACKKNIFE INFERENCE

This dialog performs jackknife inference for a specified statistic and data frame. See chapter 30 in the *Guide to Statistics* for details.

To perform jackknife inference:

Choose **Statistics>Resample>Jackknife** from the main menu. The dialog shown below appears.

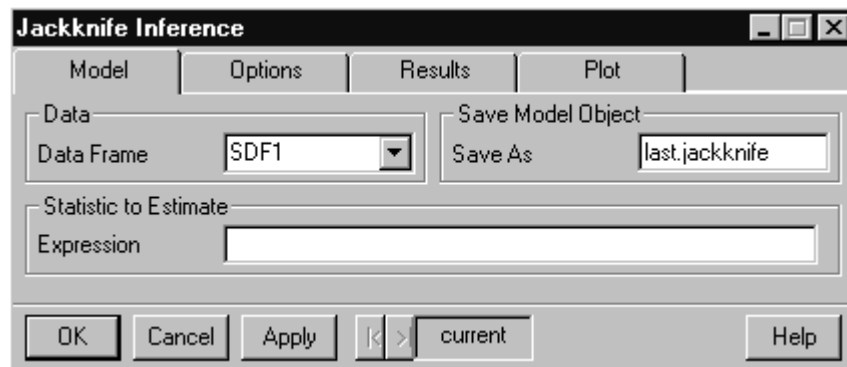


Figure 7.6: *The Jackknife Inference dialog, Model page.*

## Model Page

<b>Data</b>	<b>Data Frame</b>  Specify the data to jackknife. This may be a vector, matrix, or data frame.
<b>Statistic to Estimate</b>	<b>Expression</b>  Specify the expression describing the statistic to be jackknifed. It may be a function that accepts data as the first argument and returns a vector or matrix, or a call referring to the data that evaluates to a vector or matrix.  For example, to jackknife the regression coefficients for regressing <a href="#">Mi l eage</a>

on `Weight` in the `fuel.frame` data, use the expression `coef(lm(Mileage~Weight, fuel.frame))` and specify `fuel.frame` as the **Data Frame**. To jackknife the mean of `Mileage`, use the expression `mean(Mileage)`.

## Save Model Object

### Save As

Enter the name for the object in which to save the results of the analysis. If an object with this name already exists, its contents will be overwritten.

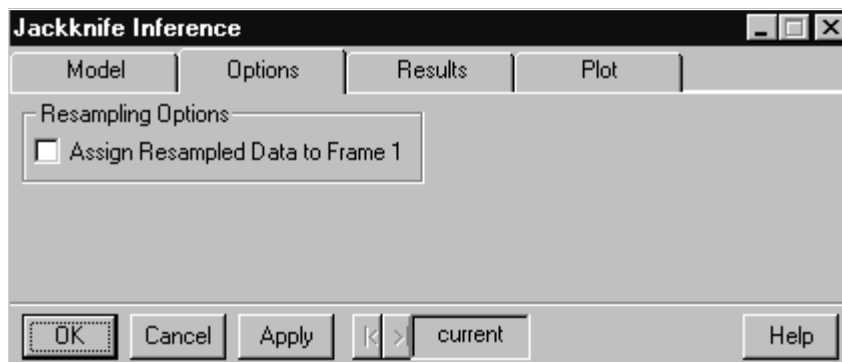


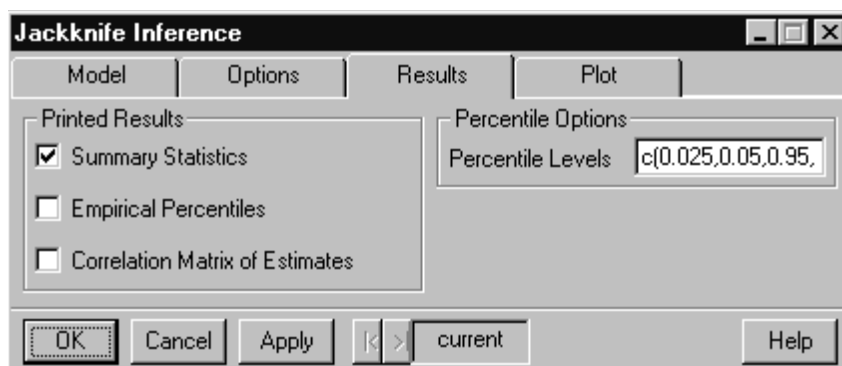
Figure 7.7: The Jackknife Inference dialog, Options page.

## Options Page

### Resampling Options

#### Assign Resampled Data to Frame 1

Check this to assign the resampled data to frame 1 as each sample is generated. See the language help for `jackknife` for details.



**Figure 7.8:** *The Jackknife Inference dialog, Results page.*

## Results Page

### Printed Results

#### Summary Statistics

Check this to print basic summaries such as the jackknife estimates of bias, mean, and standard error.

#### Empirical Percentiles

Check this to print empirical percentiles for the statistic under consideration.

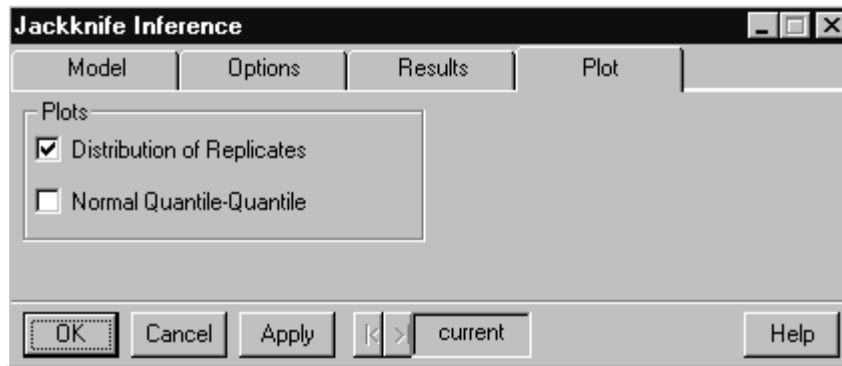
#### Correlation Matrix of Estimates

Check this to print the correlation matrix for the estimates. Note that this is only relevant if the statistic under consideration is a vector, such as a vector of regression coefficients.

### Percentile Options

#### Percentile Levels

Specify a vector of percentile levels at which to evaluate the empirical percentiles.



**Figure 7.9:** *The Jackknife Inference dialog, Plot page.*

## Plot Page

### Plots

#### Distribution of Replicates

Check this to plot the distribution of the replicates for each statistic of interest.

#### Normal Quantile-Quantile

Check this to plot a Normal quantile-quantile plot for each statistic of interest.

#### Related programming language functions:

[jackknife](#)

---

# CLUSTERING IN S-PLUS

# 8

---

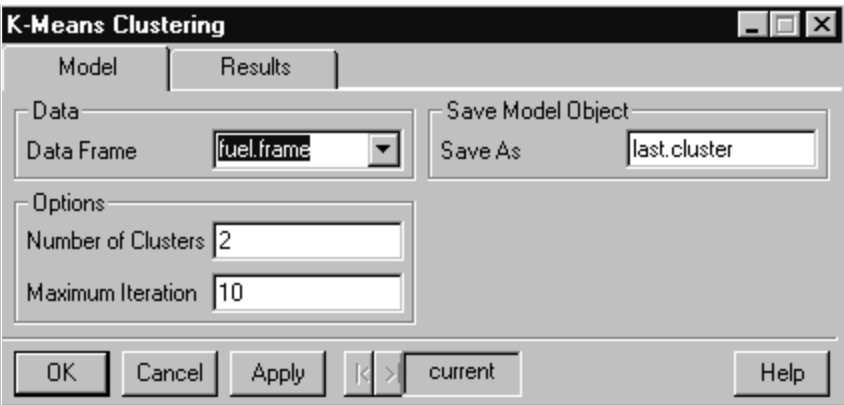
<b>K-Means Clustering</b>	<b>88</b>
Model Page	88
Results Page	89
<b>Partitioning Around Medoids</b>	<b>90</b>
Model Page	90
Results Page	92
Plot Page	93
<b>Fuzzy Partitioning</b>	<b>94</b>
Model Page	94
Results Page	96
Plot Page	97
<b>Agglomerative Hierarchical Clustering</b>	<b>98</b>
Model Page	98
Results Page	100
Plot Page	101
<b>Divisive Hierarchical Clustering</b>	<b>102</b>
Model Page	102
Results Page	104
Plot Page	105
<b>Monothetic Clustering</b>	<b>106</b>
Model Page	106
Results Page	107
Plot Page	107
<b>Compute Dissimilarities</b>	<b>109</b>

# K-MEANS CLUSTERING

This dialog performs k-means clustering. See chapter 18 in the *Guide to Statistics* for details.

To perform k-means clustering:

Choose **Statistics:Cluster Analysis:K-Means** from the main menu. The dialog shown below appears.



## Model Page

### Data

#### Data Frame

Specify the data frame. To use a subset of rows or columns, use standard S-PLUS subscripting of the data frame.

### Options

#### Number of Clusters

Specify the number of clusters to form, or a matrix of initial values for cluster centers.

#### Maximum Iteration

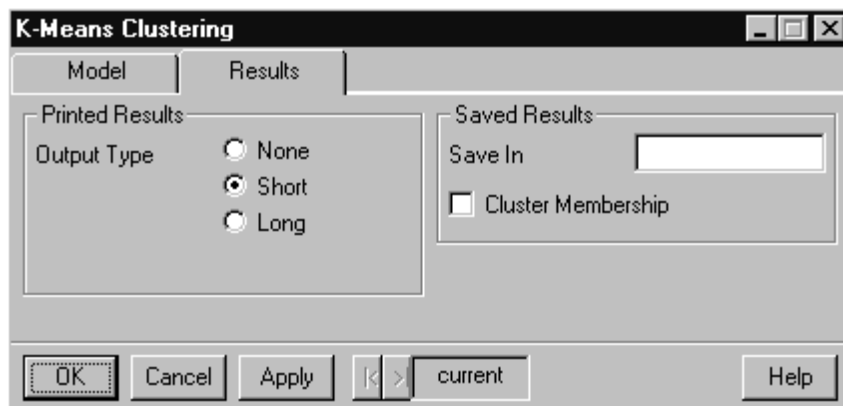
Specify the maximum number of reallocation iterations to perform.



## Save Model Object

### Save As

Enter the name for the object in which to save the results of the analysis. If an object with this name already exists, its contents will be overwritten.



## Results Page

### Printed Results

#### Output Type

Select “None” for no printed output, “Short” for a short printed summary, or “Long” for a more detailed printed summary.

### Saved Results

#### Save In

Specify the name of a data frame in which to save cluster membership if **Cluster Membership** is checked.

#### Cluster Membership

Check this to save a vector of indices giving cluster memberships in the specified data frame.

**Related programming language functions:**

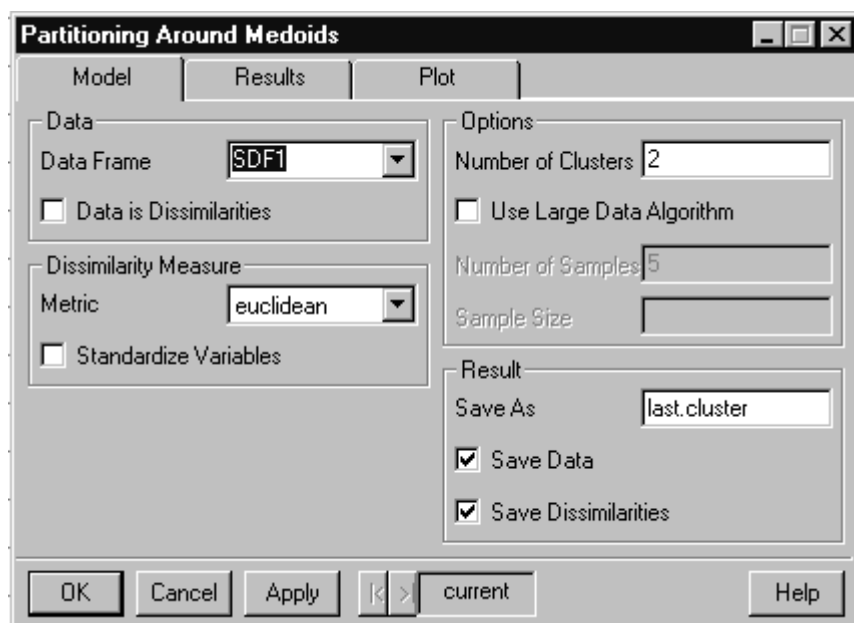
[kmeans](#)

## PARTITIONING AROUND MEDOIDS

This dialog performs partitioning around medoids. See chapter 18 in the *Guide to Statistics* for details.

To perform partitioning around medoids:

Choose **Statistics:Cluster Analysis:Partitioning Around Medoids** from the main menu. The dialog shown below appears.



### Model Page

#### Data

#### Data Frame

Specify a data frame or a `dissimilarity` object. To use a subset of rows or columns, use standard S-PLUS subscripting of the data frame.

Note that all columns of the data frame must be numeric. If non-numeric columns (e.g. factors) are present, use the **Dissimilarities** dialog to produce a `dissimilarity` object, and then use this object in clustering. The **Dissimilarities** dialog provides special options for handling factors.

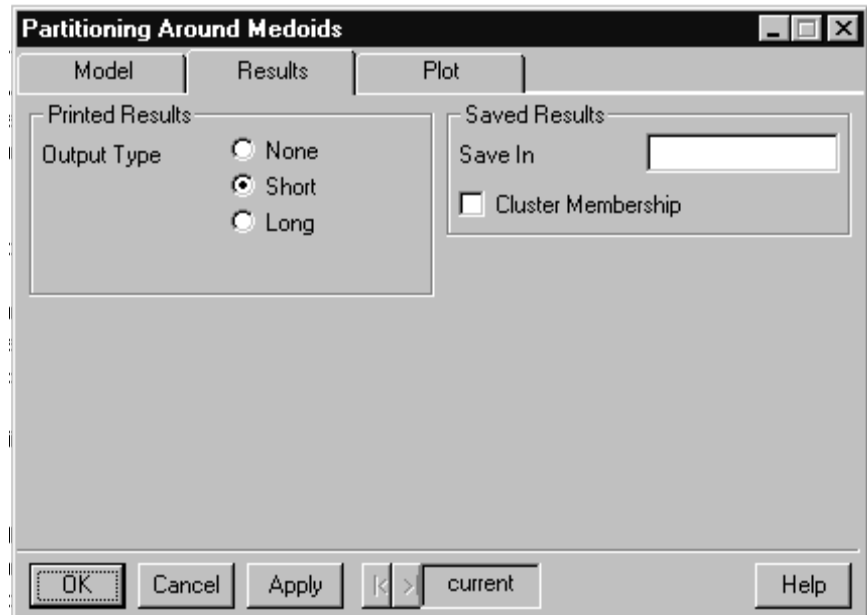
<b>Data is Dissimilarities</b>	
Check this if <b>Data Frame</b> names a <code>dissimilarity</code> object.	
<b>Dissimilarity Measure</b>	<p><b>Metric</b></p> <p>Select the metric to be used for calculating dissimilarities between objects. The available options are <code>"euclidean"</code> and <code>"manhattan"</code>. Euclidean distances are root sum-of-squares of differences, and manhattan distances are the sum of absolute differences. If <b>Data Frame</b> is already a dissimilarity matrix, then this argument will be ignored.</p> <p><b>Standardize Variables</b></p> <p>Check this to standardize each data column by subtracting the variable's mean value and dividing by the variable's mean absolute deviation. If <b>Data Frame</b> is already a dissimilarity matrix, then this argument will be ignored.</p>
<b>Options</b>	<p><b>Number of Clusters</b></p> <p>Specify the number of clusters to form.</p> <p><b>Use Large Data Algorithm</b></p> <p>Check this box to use the "Clustering Large Applications" algorithm. This algorithm considers data subsets of fixed size, so that the overall time and storage requirements become linear in the total number of objects, rather than quadratic. Note that this algorithm is not available for <b>dissimilarity</b> objects.</p> <p><b>Number of Samples</b></p> <p>Number of subsets to draw from the data when using the large data algorithm. See the help file for <b>clara</b> for details.</p> <p><b>Sample Size</b></p> <p>Size of each subset when using the large data algorithm. See the help file for <b>clara</b> for details.</p>
<b>Save Model Object</b>	<p><b>Save As</b></p> <p>Enter the name for the object in which to save the results of the analysis. If an object with this name already exists, its contents will be overwritten.</p>

## Save Data

Check this box to store a copy of the data in the model object. This is necessary if you wish to produce a clusplot for the model.

## Save Dissimilarities

Check this box to store a copy of the dissimilarities in the model object. This is necessary if you wish to produce a clusplot for the model.



## Results Page

### Printed Results      Output Type

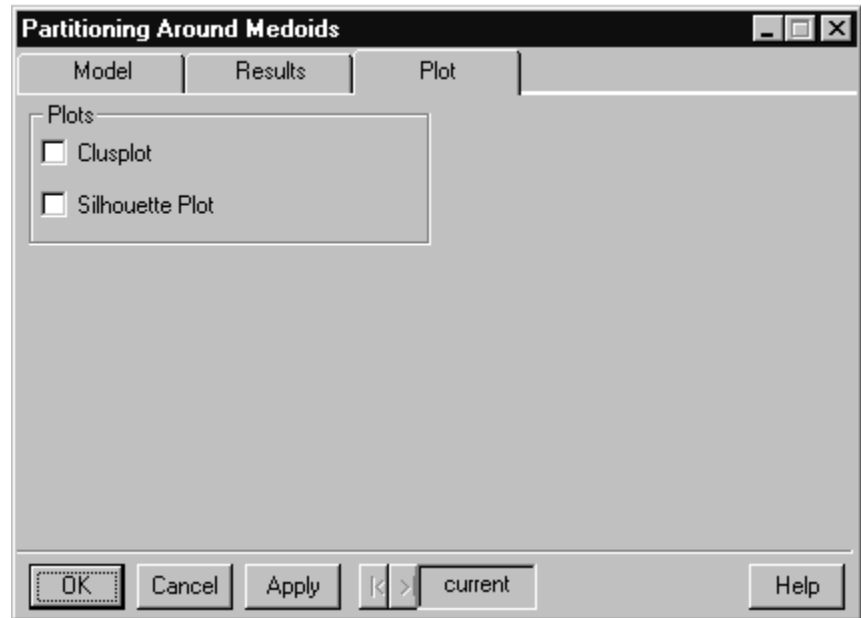
Select “None” for no printed output, “Short” for a short printed summary, or “Long” for a more detailed printed summary.

### Save Results      Save In

Specify the name of a data frame in which to save cluster membership if **Cluster Membership** is checked.

## Cluster Membership

Check this to save a vector of indices giving cluster memberships in the specified data frame.



## Plot Page

### Plots

#### Clusplot

Check this to create a clusplot for the clustering.

#### Silhouette Plot

Check this to create a silhouette plot for the clustering.

**Related programming language functions:**

[pam](#), [clara](#)

## FUZZY PARTITIONING

This dialog performs fuzzy partitioning. See chapter 18 in the *Guide to Statistics* for details.

To perform fuzzy partitioning:

Choose **Statistics:Cluster Analysis:Fuzzy Partitioning** from the main menu. The dialog shown below appears.

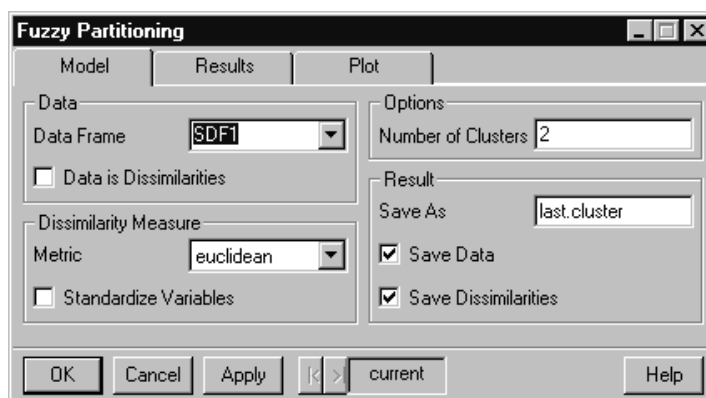


Figure 8.1: The Fuzzy Partitioning dialog, Model page.

### Model Page

#### Data

#### Data Frame

Specify a data frame or a `dis` object. To use a subset of rows or columns, use standard S-PLUS subscripting of the data frame.

Note that all columns of the data frame must be numeric. If non-numeric columns (e.g. factors) are present, use the **Dissimilarities** dialog to produce a `dis` object, and then use this object in clustering. The **Dissimilarities** dialog provides special options for handling factors.

---

## Data is Dissimilarities

Check this if **Data Frame** names a `dissimilarity` object.

### Dissimilarity Measure

#### Metric

Select the metric to be used for calculating dissimilarities between objects. The available options are "`euclidean`" and "`manhattan`". Euclidean distances are root sum-of-squares of differences, and manhattan distances are the sum of absolute differences. If **Data Frame** is already a dissimilarity matrix, then this argument will be ignored.

#### Standardize Variables

Check this to standardize each data column by subtracting the variable's mean value and dividing by the variable's mean absolute deviation. If **Data Frame** is already a dissimilarity matrix, then this argument will be ignored.

### Options

#### Number of Clusters

Specify the number of clusters to form.

### Result

#### Save As

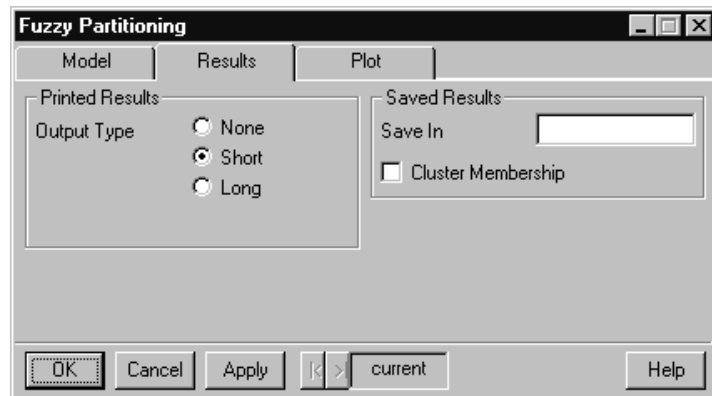
Enter the name for the object in which to save the results of the analysis. If an object with this name already exists, its contents will be overwritten.

#### Save Data

Check this box to store a copy of the data in the model object. This is necessary if you wish to produce a `clusplot` for the model.

#### Save Dissimilarities

Check this box to store a copy of the dissimilarities in the model object. This is necessary if you wish to produce a `clusplot` for the model.



**Figure 8.2:** *The Fuzzy Partitioning dialog, Results page.*

## Results Page

### Printed Results

#### Output Type

Select “None” for no printed output, “Short” for a short printed summary, or “Long” for a more detailed printed summary.

### Saved Results

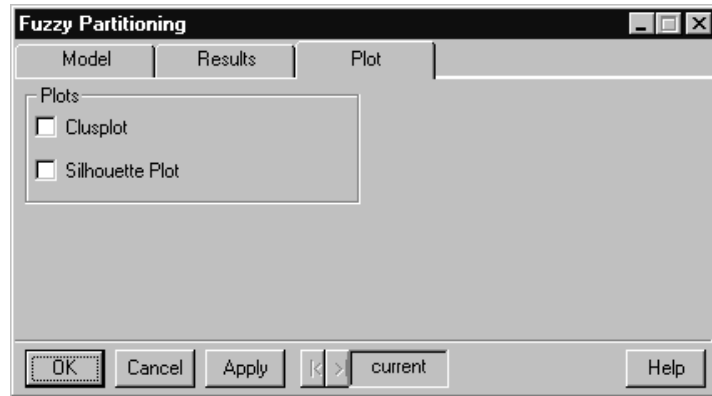
#### Save In

Specify the name of a data frame in which to save cluster membership if **Cluster Membership** is checked.

#### Cluster Membership

Check this to save a vector of indices giving cluster memberships in the specified data frame.





**Figure 8.3:** *The Fuzzy Partitioning dialog, Plot page.*

## Plot Page

### Plots

#### Clusplot

Check this to create a clusplot for the clustering.

#### Silhouette Plot

Check this to create a silhouette plot for the clustering.

**Related programming language functions:**

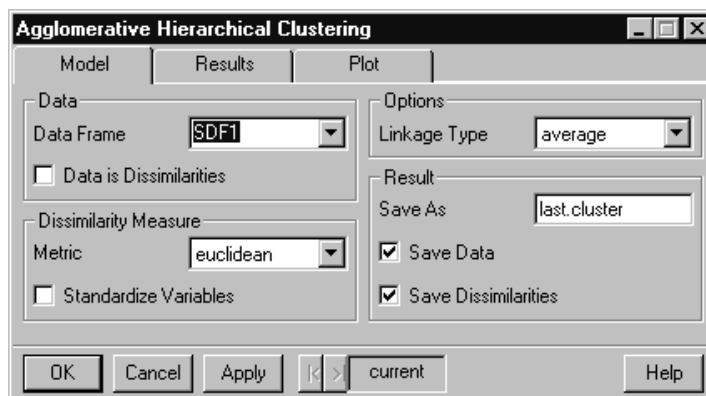
[fanny](#)

## AGGLOMERATIVE HIERARCHICAL CLUSTERING

This dialog performs agglomerative hierarchical clustering. See chapter 18 in the *Guide to Statistics* for details.

To perform agglomerative hierarchical clustering:

Choose **Statistics:Cluster Analysis:Agglomerative Hierarchical** from the main menu. The dialog shown below appears.



**Figure 8.4:** *The Agglomerative Hierarchical Clustering dialog, Model page.*

### Model Page

#### Data

#### Data Frame

Specify a data frame or a `dis`similarity object. To use a subset of rows or columns, use standard S-PLUS subscripting of the data frame.

Note that all columns of the data frame must be numeric. If non-numeric columns (e.g. factors) are present, use the **Dissimilarities** dialog to produce a `dis`similarity object, and then use this object in clustering. The **Dissimilarities** dialog provides special options for handling factors.

## Data is Dissimilarities

Check this if **Data Frame** names a `dissimilarity` object.

### Dissimilarity Measure

#### Metric

Select the metric to be used for calculating dissimilarities between objects. The available options are `"euclidean"` and `"manhattan"`. Euclidean distances are root sum-of-squares of differences, and manhattan distances are the sum of absolute differences. If **Data Frame** is already a dissimilarity matrix, then this argument will be ignored.

#### Standardize Variables

Check this to standardize each data column by subtracting the variable's mean value and dividing by the variable's mean absolute deviation. If **Data Frame** is already a dissimilarity matrix, then this argument will be ignored.

### Options

#### Linkage Type

Specify the linkage type. The three methods implemented are `"average"`, `"complete"`, `"single"`, `"ward"`, and `"weighted"` linkage.

### Result

#### Save As

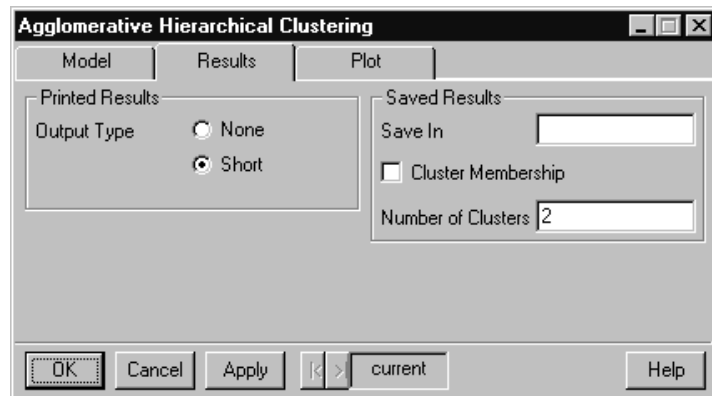
Enter the name for the object in which to save the results of the analysis. If an object with this name already exists, its contents will be overwritten.

#### Save Data

Check this box to store a copy of the data in the model object.

#### Save Dissimilarities

Check this box to store a copy of the dissimilarities in the model object.



**Figure 8.5:** *The Agglomerative Hierarchical Clustering dialog, Results page.*

## Results Page

### Printed Results

#### Output Type

Select “None” for no printed output or “Short” for a short printed summary.

### Saved Results

#### Save In

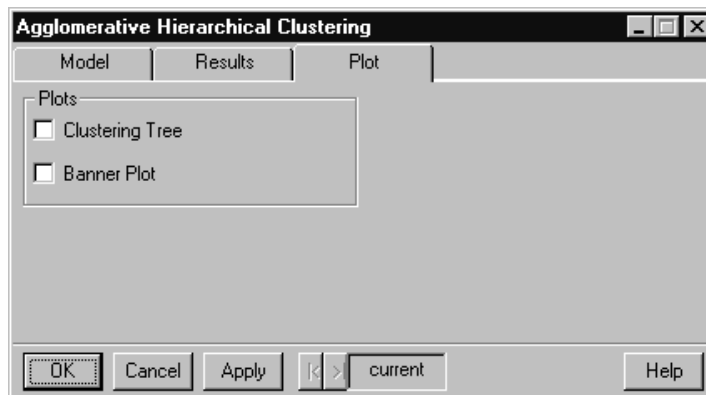
Specify the name of a data frame in which to save cluster membership if **Cluster Membership** is checked.

#### Cluster Membership

Check this to save a vector of indices giving cluster memberships in the specified data frame.

#### Number of Clusters

Specify the number of clusters to form when generating cluster membership indices.



**Figure 8.6:** *The Agglomerative Hierarchical Clustering dialog, Plot page.*

## Plot Page

### Plots

#### Clustering Tree

Check this to create a clustering tree plot.

#### Banner Plot

Check this to create a banner plot.

**Related programming language functions:**

[agnes](#)

## DIVISIVE HIERARCHICAL CLUSTERING

This dialog performs divisive hierarchical clustering. See chapter 18 in the *Guide to Statistics* for details.

To perform divisive hierarchical clustering:

Choose **Statistics:Cluster Analysis:Divisive Hierarchical** from the main menu. The dialog shown below appears.

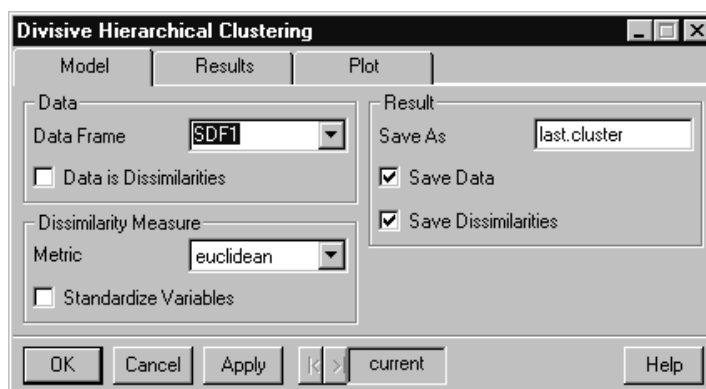


Figure 8.7: The Divisive Hierarchical Clustering dialog, Model page.

### Model Page

#### Data

#### Data Frame

Specify a data frame or a [dissimilarity](#) object. To use a subset of rows or columns, use standard S-PLUS subscripting of the data frame.

Note that all columns of the data frame must be numeric. If non-numeric columns (e.g. factors) are present, use the **Dissimilarities** dialog to produce a [dissimilarity](#) object, and then use this object in clustering. The **Dissimilarities** dialog provides special options for handling factors.

---

	<b>Data is Dissimilarities</b>
	Check this if <b>Data Frame</b> names a <code>dissimilarity</code> object.
<b>Dissimilarity Measure</b>	<b>Metric</b>  Select the metric to be used for calculating dissimilarities between objects. The available options are " <code>euclidean</code> " and " <code>manhattan</code> ". Euclidean distances are root sum-of-squares of differences, and manhattan distances are the sum of absolute differences. If <b>Data Frame</b> is already a dissimilarity matrix, then this argument will be ignored.  <b>Standardize Variables</b>  Check this to standardize each data column by subtracting the variable's mean value and dividing by the variable's mean absolute deviation. If <b>Data Frame</b> is already a dissimilarity matrix, then this argument will be ignored.
<b>Result</b>	<b>Save As</b>  Enter the name for the object in which to save the results of the analysis. If an object with this name already exists, its contents will be overwritten.  <b>Save Data</b>  Check this box to store a copy of the data in the model object.  <b>Save Dissimilarities</b>  Check this box to store a copy of the dissimilarities in the model object.

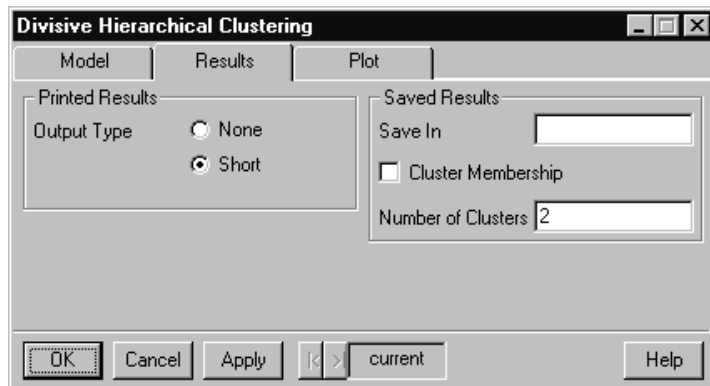


Figure 8.8: *The Divisive Hierarchical Clustering dialog, Results page.*

## Results Page

### Printed Results

#### Output Type

Select “None” for no printed output or “Short” for a short printed summary.

### Saved Results

#### Save In

Specify the name of a data frame in which to save cluster membership if **Cluster Membership** is checked.

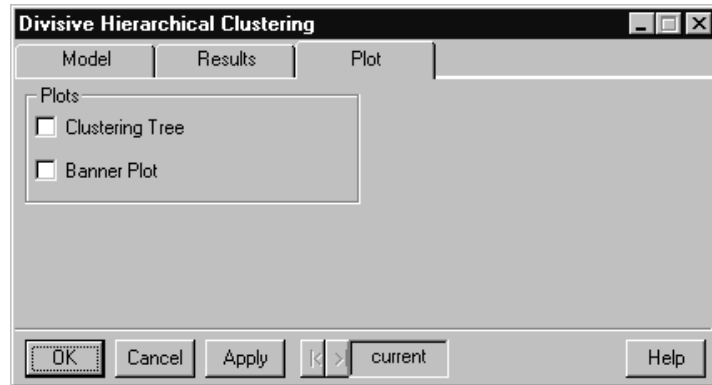
#### Cluster Membership

Check this to save a vector of indices giving cluster memberships in the specified data frame.

#### Number of Clusters

Specify the number of clusters to form when generating cluster membership indices.





**Figure 8.9:** *The Divisive Hierarchical Clustering dialog, Plot page.*

## Plot Page

### Plots

#### Clustering Tree

Check this to create a clustering tree plot.

#### Banner Plot

Check this to create a banner plot.

**Related programming language functions:**

[di ana](#)

# MONOTHETIC CLUSTERING

This dialog performs monothetic clustering. This clustering technique may be used to partition data when all variables are binary. See chapter 18 in the *Guide to Statistics* for details.

To perform monothetic clustering:

Choose **Statistics:Cluster Analysis:Monothetic (Binary Variables)** from the main menu. The dialog shown below appears.

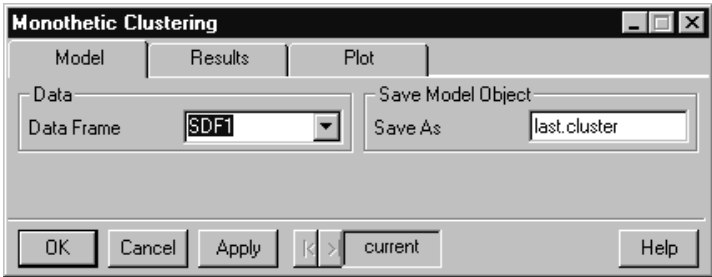


Figure 8.10: The Monothetic Clustering dialog, Model page.

## Model Page

### Data

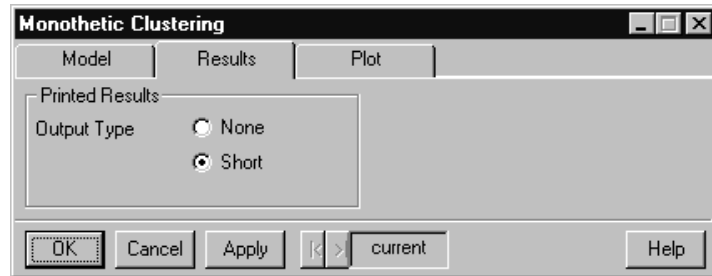
#### Data Frame

Specify the data frame. For monothetic analysis, all variables must be binary. A limited number of missing values (NAs) is allowed. Every observation must have at least one value different from NA. No variable should have half of its values missing. There must be at least one variable which has no missing values. A variable with all its non-missing values identical, is not allowed.

### Save Model Object

#### Save As

Enter the name for the object in which to save the results of the analysis. If an object with this name already exists, its contents will be overwritten.

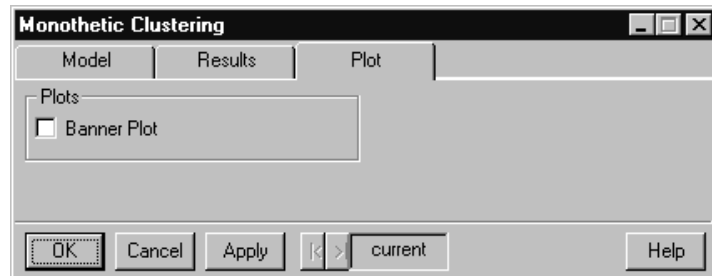


**Figure 8.11:** *The Monothetic Clustering dialog, Results page.*

## Results Page

### Printed Results      Output Type

Select “None” for no printed output or “Short” for a short printed summary.



**Figure 8.12:** *Monothetic Clustering dialog, Plot page.*

## Plot Page

### Plots      Banner Plot

Check this to create a banner plot.

**Related programming language functions:**

[mona](#)

# COMPUTE DISSIMILARITIES

This dialog calculates dissimilarities for a data frame. Different types of variables (e.g. numeric and factor) are handled in appropriate manners. See chapter 18 in the *Guide to Statistics* for details.

To calculate dissimilarities:

Choose **Statistics:Multivariate:Cluster:Dissimilarities** from the main menu. The dialog shown below appears.

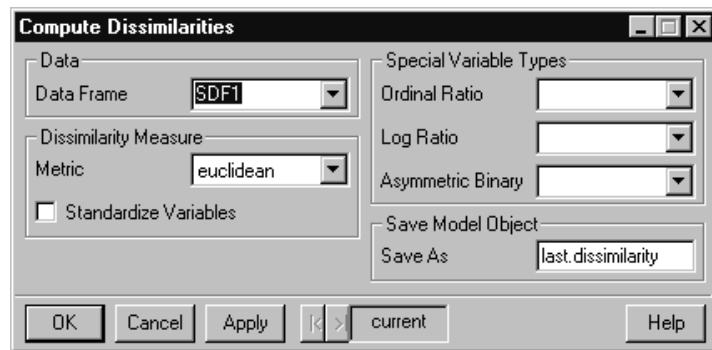


Figure 8.13: *The Compute Dissimilarities dialog.*

## Data

### Data Frame

Specify the data frame.

## Dissimilarity Measure

### Metric

Select the metric to be used for calculating dissimilarities between objects. The available options are "euclidean" and "manhattan". Euclidean distances are root sum-of-squares of differences, and manhattan distances are the sum of absolute differences.

### Standardize Variables

Check this to standardize each data column by subtracting the variable's mean value and dividing by the variable's mean absolute deviation.

**Special Variable  
Types****Ordinal Ratio**

Select variables to be treated as ordinal ratio variables.

**Log Ratio**

Select variables to be treated as log ratio variables.

**Asymmetric Binary**

Select variables to be treated as asymmetric binary variables.

**Save Model  
Object****Save As**

Enter the name for the object in which to save the results of the analysis. If an object with this name already exists, its contents will be overwritten.

**Related programming language functions:**

[dai sy](#)

# CREATING HTML OUTPUT

# 9

---

<b>Tables</b>	<b>112</b>
<b>Text</b>	<b>113</b>
<b>Graphs</b>	<b>114</b>

S-PLUS provides a variety of tools for generating HTML output. This chapter discusses how to generate HTML tables, save preformatted text output, and save graphs with HTML references.

## TABLES

The `html.table` function may be used to generate a vector of character strings representing a vector, matrix, or data frame as an HTML table. The vector will contain one string for each line of HTML. This may be written to a file by specifying the `file` argument, or may be manipulated and later written to a file using the `write` function.

For example, we can create a file `catalyst.htm` containing the `catalyst` data frame using:

```
> html.table(catalyst, file="catalyst.htm")
```

In addition to accepting a vector, matrix, or data frame, the `html.table` function will accept a simple list with such structures as components of the list. It will then produce a sequence of tables with the list component names encoded as table captions. For example:

```
> my.results<-list("Regression Coefficients" =
+   coef(lm(Mileage~Weight,
+   fuel.frame))), "Correlations"=cor(fuel.frame[, 1:3]))

> html.table(my.results, file="my.htm")
```

The `html.table` function accepts any of the arguments to `format`, allowing specification of formatting details such as the number of digits displayed. In addition, `append` controls whether output is appended to the specified `file` or the file is overwritten. The `append` argument is also available in the `write` function, which is useful for interspersing `html.table` output and descriptive text:

```
> write("<H3> S-PLUS Code for the above </H3>
Continue string: <P> Put code here </P>",
+   file="my.htm", append=T)
```

Additional arguments to `html.table` are described in the function's help file.

Note that `html.table` is designed to work with the previously mentioned data structures. For other structures such as functions, calls, and objects with specific `print` methods, the results of `html.table` may not be satisfactory. Instead, the object may be printed as preformatted text and embedded in the HTML page.



---

## TEXT

The `si nk` function may be used to direct S-PLUS text output to an HTML file. The preformatted output may be interspersed with the HTML markup tag `<PRE>` to denote that it is preformatted output. Additional textual description and HTML markup tags may be interspersed with the S-PLUS output using `cat`.

```
> si nk("my. htm")  
  
> cat("<H3> Linear Model Resul ts </H3> \n")  
  
> cat("<PRE>")  
  
> summary(lm(Mi l eage~Wei ght, fuel . frame))  
  
> cat("</PRE>")  
  
> si nk()
```

The `paste` and `deparse` functions are useful for constructing strings to display with `cat`. See their help files for details.

## GRAPHS

The two steps involved in embedding an S-PLUS graph in an HTML page are exporting the graph in a format such as GIF or JPG which is viewable with a web browser, and placing an `<IMG>` tag in the HTML file describing the location of the image.

Use the `export.graph` command to export a graph to a specific file:

```
> graphsheet(Name="MyGraph")  
  
> xyplot(Mileage~Weight, fuel.frame)  
  
> export.graph(Name="MyGraph", FileName="my.gif")
```

Use `sink` and `cat` to place an `<IMG>` tag in an HTML file. Note the use of `\` to include quotation marks in the text:

```
> sink("my.htm", append=T)  
  
> cat("<IMG SRC=\"my.gif\">")  
  
> sink()
```

# TYPE III SUM OF SQUARES AND ADJUSTED MEANS

# 10

Researchers implementing an experimental design frequently lose experimental units and find themselves with unbalanced, but complete, data. The data is unbalanced in that the number of replications is not constant for each treatment combination; the data is complete in that at least one experimental unit exists for each treatment combination. In this type of circumstance, an experimenter may find the hypotheses tested by Type III sum of squares are of more interest than those tested by Type I (sequential) sum of squares, and the adjusted means of more interest than unadjusted means. New options to the `lm` and `aov` object methods, `anova.lm`, `summary.aov`, and `model.tables.aov` will give the Type III sum of squares and the adjusted (marginal) means. For `anova` and `summary`, the new argument `ssType` can be 1 or 3, with `ssType=1` as the default; `model.tables` has the new option `"adj.means"`, for the existing argument type. An example is given to demonstrate the new capabilities of these in an analysis of a designed experiment.

The fat-surfactant example is taken from Milliken and Johnson (1984, p. 166), where they analyze an unbalanced randomized block factorial design. Here, the specific volume of bread loaves baked from dough mixed from each of nine Fat and Surfactant treatment combinations is measured. The experimenters blocked on four flour types. Ten loaves had to be removed from the experiment, but at least one loaf existed for each Fat  $\times$  Surfactant combination and all marginal means are estimable so the Type III hypotheses are testable.

The over-parameterized model is:

$$\mu_{ijk} = \mu + b_i + f_j + s_k + (fs)_{jk} ,$$

for  $i=1,\dots,4$ ,  $j=1,2,3$ , and  $k=1,2,3$ . Because the data are unbalanced the Type III sum of squares for Flour, Fat and Surfactant test a more useful hypothesis than the Type I. Specifically, the Type III hypotheses are that the marginal means are equal:

$$H_{\text{Flour}} : \bar{\mu}_{1..} = \bar{\mu}_{2..} = \bar{\mu}_{3..} = \bar{\mu}_{4..}$$

$$H_{\text{Fat}} : \bar{\mu}_{.1.} = \bar{\mu}_{.2.} = \bar{\mu}_{.3.}$$

$$H_{\text{Surfactant}} : \bar{\mu}_{..1} = \bar{\mu}_{..2} = \bar{\mu}_{..3} ,$$

where

$$\bar{\mu}_{i..} = \frac{\sum_{j,k} \mu_{ijk}}{3 \cdot 3}$$

$$\bar{\mu}_{.j.} = \frac{\sum_{i,k} \mu_{ijk}}{4 \cdot 3}$$

$$\bar{\mu}_{..k} = \frac{\sum_{ij} \mu_{ijk}}{4 \cdot 3} .$$

The hypotheses tested by the Type I sum of squares are not easily interpreted since they are dependent on the order each term is specified the formula and involve the cell replications (which can be viewed as random variables when there are random drop-outs). Moreover, the hypothesis tested by the blocking term, Flour, involves parameters of the Fat, Flour and Fat  $\times$  Flour terms.

## ANOVA Tables

The ANOVA tables for both Type I and Type III sum of squares are given below for comparison. Using the Type III sum of squares we see that the block effect, Flour, is significant as is Fat, but Surfactant is not at, say, a test size of  $\alpha = 0.05$ . However, in the presence of a significant interaction, the test of the marginal means probably has little meaning for Fat and Surfactant.

```
> Baki ng. aov<-aov(Speci fi c. Vol ~ Flour + Fat * Surfactant,
+ data = Baki ng, contrasts=list(Fl our=contr.sum(4),
+ Fat=contr.sum(3), Surfactant=contr.sum(3)))
> anova(Baki ng. aov)
```

Anal ysi s of Vari ance Tabl e

Response: Speci fi c. Vol

Terms added sequentially (first to last)

	Df	Sum of Sq	Mean Sq	F Value	Pr(F)
Fl our	3	6.39310	2.131033	12.88269	0.0002587
Fat	2	10.33042	5.165208	31.22514	0.0000069
Surfactant	2	0.15725	0.078625	0.47531	0.6313678
Fat: Surfactant	4	5.63876	1.409691	8.52198	0.0010569
Resi dual s	14	2.31586	0.165418		

```
> anova(Baki ng. aov, ssType=3)
```

Anal ysi s of Vari ance Tabl e

Response: Speci fi c. Vol

Type III Sum of Squares

	Df	Sum. of. Sq	Mean. Sq	F. Val ue	Pr. F.
Fl our	3	8.69081	2.896937	17.51280	0.00005181
Fat	2	10.11785	5.058925	30.58263	0.00000778
Surfactant	2	0.99721	0.498605	3.01421	0.08153989
Fat: Surfactant	4	5.63876	1.409691	8.52198	0.00105692
Resi dual s	14	2.31586	0.165418		

## Adjusted Means

The adjusted (marginal) means given below estimate the means given in the Type III hypotheses for Flour, Fat and Surfactant. The means for Flour × Surfactant for the over-parameterized model are

$$\bar{\mu}_{\cdot jk} = \frac{\sum_i \mu_{ijk}}{4}$$

Interestingly, these means are still estimable even though not all Flour × Surfactant × Flour combinations were observed.

```
> model.tables(Baking.aov, type="adj.means")
```

```
Tables of adjusted means
```

```
Grand mean
```

```
6.633281
```

```
se 0.084599
```

```
N 26.000000
```

```
Flour
```

```
1 2 3 4
```

```
7.3020 5.7073 6.9815 6.5423
```

```
se 0.1995 0.1467 0.1621 0.1785
```

```
rep 5.0000 8.0000 7.0000 6.0000
```

```
Fat
```

```
1 2 3
```

```
5.8502 6.5771 7.4725
```

```
se 0.1365 0.1477 0.1565
```

```
rep 9.0000 9.0000 8.0000
```

```
Surfactant
```

```
1 2 3
```

```
6.3960 6.5999 6.9039
```

```
se 0.1502 0.1432 0.1473
```

```
rep 8.0000 9.0000 9.0000
```

```
Fat: Surfactant
```

```
Dim 1 : Fat
```

```
Dim 2 : Surfactant
```

```
1 2 3
```

```
1 5.5364 5.8913 6.1229
```

```
se 0.2404 0.2392 0.2414
```

```
rep 3.0000 3.0000 3.0000
```

```
2 7.0229 6.7085 6.0000
```

```
se 0.2414 0.3006 0.2034
```

```
rep 3.0000 2.0000 4.0000
```

```
3 6.6286 7.2000 8.5889
```

```
se 0.3007 0.2034 0.3001
```

```
rep 2.0000 4.0000 2.0000
```

## Multiple Comparisons

The F-statistic for the Fat  $\times$  Surfactant interaction in the Type III ANOVA table is significant so the tests for the marginal means for Fat and Surfactant have little meaning. We can, however, use `multcomp` to find all pairwise comparisons of the mean Fat levels for each level of Surfactant, and those for Surfactant for each level of Fat.

---

```
> multi comp(Baking. aov, focus="Fat",
+ adjust=list(Surfactant=seq(3)))
```

95 % simultaneous confidence intervals for specified linear combinations, by the Sidak method

critical point: 3.2117

response variable: Flour

intervals excluding 0 are flagged by '\*\*\*\*'

	Estimate	Std. Error	Lower Bound	Upper Bound	
1. adj 1-2. adj 1	-1.490	0.344	-2.590	-0.381	****
1. adj 1-3. adj 1	-1.090	0.377	-2.300	0.120	
2. adj 1-3. adj 1	0.394	0.394	-0.872	1.660	
1. adj 2-2. adj 2	-0.817	0.390	-2.070	0.434	
1. adj 2-3. adj 2	-1.310	0.314	-2.320	-0.300	****
2. adj 2-3. adj 2	-0.492	0.363	-1.660	0.674	
1. adj 3-2. adj 3	0.123	0.316	-0.891	1.140	
1. adj 3-3. adj 3	-2.470	0.378	-3.680	-1.250	****
2. adj 3-3. adj 3	-2.590	0.363	-3.750	-1.420	****

```
> multi comp(Baking. aov, focus="Surfactant",
+ adjust=list(Fat=seq(3)))
```

95 % simultaneous confidence intervals for specified linear combinations, by the Sidak method

critical point: 3.2117

response variable: Flour

intervals excluding 0 are flagged by '\*\*\*\*'

	Estimate	Std. Error	Lower Bound	Upper Bound	
1. adj 1-2. adj 1	-0.355	0.341	-1.45000	0.740	
1. adj 1-3. adj 1	-0.587	0.344	-1.69000	0.519	
2. adj 1-3. adj 1	-0.232	0.342	-1.33000	0.868	
1. adj 2-2. adj 2	0.314	0.377	-0.89700	1.530	
1. adj 2-3. adj 2	1.020	0.316	0.00922	2.040	****
2. adj 2-3. adj 2	0.708	0.363	-0.45700	1.870	
1. adj 3-2. adj 3	-0.571	0.363	-1.74000	0.594	
1. adj 3-3. adj 3	-1.960	0.427	-3.33000	-0.590	****
2. adj 3-3. adj 3	-1.390	0.363	-2.55000	-0.225	****

The levels for Fat and Surfactant factors are both labeled 1, 2, and 3 so the row labels in the multi comp tables require explanation. For the first table, the label 1. adj 1-2. adj 1 refers to the difference between levels 1 and 2 of Fat (the focus variable) at level 1 of Surfactant (the adjust variable), whereas for the second table it is the difference between levels 1 and 2 of Surfactant at level 1 of Fat. The reader can verify that the table of differences reported by multi comp are the differences in the adjusted means for Fat: Surfactant reported by model . tables. Significant differences are flagged with '\*\*\*\*'. As a result of the of Surfactant and Fat interaction, the

F test for the equivalence of the Surfactant marginal means is not significant, but there exists significant differences between the mean of Surfactant levels 1-3 at a Fat level of 2 and between the means Surfactant levels 1-3 and 2-3 at a Fat level of 3.

## Estimable Functions

The Type I and Type III estimable functions for the over-parameterized model show the linear combinations of the over-parameterized model parameters tested by each sum of squares. The Type I estimable functions can be obtained by performing row reductions on the cross products of the model matrix,  $X^tX$ , that reduce it to upper triangular with each nonzero row divided by its diagonal (SAS Technical Report R-101, 1978).

```
> round(L, 4)
```

	L2	L3	L4	L6	L7	L9	L10	L12	L13	L15	L16
(Intercept)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0	0	0	0
Flour. 1	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0	0	0	0
Flour. 2	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0	0	0	0
Flour. 3	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0	0	0	0
Flour. 4	-1.0000	-1.0000	-1.0000	0.0000	0.0000	0.0000	0.0000	0	0	0	0
Fat. 1	0.0667	-0.0833	0.0952	1.0000	0.0000	0.0000	0.0000	0	0	0	0
Fat. 2	-0.3000	-0.1250	-0.2143	0.0000	1.0000	0.0000	0.0000	0	0	0	0
Fat. 3	0.2333	0.2083	0.1190	-1.0000	-1.0000	0.0000	0.0000	0	0	0	0
Surfactant. 1	0.2333	0.2083	0.1190	0.1152	0.1338	1.0000	0.0000	0	0	0	0
Surfactant. 2	-0.1000	-0.2500	-0.2143	-0.1966	-0.3235	0.0000	1.0000	0	0	0	0
Surfactant. 3	-0.1333	0.0417	0.0952	0.0814	0.1896	-1.0000	-1.0000	0	0	0	0
Fat. 1: Surfactant. 1	0.2000	0.1250	0.1429	0.3531	0.0359	0.3507	0.0037	1	0	0	0
Fat. 2: Surfactant. 1	-0.1667	-0.0417	-0.0238	-0.0060	0.3250	0.4242	0.0760	0	1	0	0
Fat. 3: Surfactant. 1	0.2000	0.1250	0.0000	-0.2319	-0.2271	0.2251	-0.0797	-1	-1	0	0
Fat. 1: Surfactant. 2	0.0333	-0.1667	-0.0238	0.3167	-0.0060	-0.0149	0.3499	0	0	1	0
Fat. 2: Surfactant. 2	-0.1667	-0.0417	-0.1667	0.0049	0.2034	0.0190	0.2971	0	0	0	1
Fat. 3: Surfactant. 2	0.0333	-0.0417	-0.0238	-0.5182	-0.5209	-0.0041	0.3530	0	0	-1	-1
Fat. 1: Surfactant. 3	-0.1667	-0.0417	-0.0238	0.3302	-0.0299	-0.3358	-0.3536	-1	0	-1	0
Fat. 2: Surfactant. 3	0.0333	-0.0417	-0.0238	0.0011	0.4716	-0.4432	-0.3731	0	-1	0	-1
Fat. 3: Surfactant. 3	0.0000	0.1250	0.1429	-0.2499	-0.2520	-0.2210	-0.2733	1	1	1	1

The columns labeled L2, L3, and L4 are for the Flour hypothesis; L6 and L7 are for the Fat hypothesis; L9 and L10 are for the Surfactant hypothesis; and L12, L13, L15, and L16 are for the Fat  $\times$  Surfactant hypothesis. In contrast, the Type III estimable functions can be obtained from the generating set  $(X^tX)^*(X^tX)$ , where  $(X^tX)^*$  is the g-2 inverse of the cross product matrix, (Kennedy and Gentle, 1980, p. 396) and perform the steps outlined in the SAS/STAT User's Guide (1990, pp. 120-121).



---

```

> round(L3, 4)
              L2 L3 L4          L6          L7          L9          L10 L12 L13 L15 L16
(Intercept)  0  0  0  0.0000  0.0000  0.0000  0.0000  0  0  0  0
Flour. 1      1  0  0  0.0000  0.0000  0.0000  0.0000  0  0  0  0
Flour. 2      0  1  0  0.0000  0.0000  0.0000  0.0000  0  0  0  0
Flour. 3      0  0  1  0.0000  0.0000  0.0000  0.0000  0  0  0  0
Flour. 4     -1 -1 -1  0.0000  0.0000  0.0000  0.0000  0  0  0  0
Fat. 1        0  0  0  1.0000  0.0000  0.0000  0.0000  0  0  0  0
Fat. 2        0  0  0  0.0000  1.0000  0.0000  0.0000  0  0  0  0
Fat. 3        0  0  0 -1.0000 -1.0000  0.0000  0.0000  0  0  0  0
Surfactant. 1  0  0  0  0.0000  0.0000  1.0000  0.0000  0  0  0  0
Surfactant. 2  0  0  0  0.0000  0.0000  0.0000  1.0000  0  0  0  0
Surfactant. 3  0  0  0  0.0000  0.0000 -1.0000 -1.0000  0  0  0  0
Fat. 1: Surfactant. 1  0  0  0  0.3333  0.0000  0.3333  0.0000  1  0  0  0
Fat. 2: Surfactant. 1  0  0  0  0.0000  0.3333  0.3333  0.0000  0  1  0  0
Fat. 3: Surfactant. 1  0  0  0 -0.3333 -0.3333  0.3333  0.0000 -1 -1  0  0
Fat. 1: Surfactant. 2  0  0  0  0.3333  0.0000  0.0000  0.3333  0  0  1  0
Fat. 2: Surfactant. 2  0  0  0  0.0000  0.3333  0.0000  0.3333  0  0  0  1
Fat. 3: Surfactant. 2  0  0  0 -0.3333 -0.3333  0.0000  0.3333  0  0 -1 -1
Fat. 1: Surfactant. 3  0  0  0  0.3333  0.0000 -0.3333 -0.3333 -1  0 -1  0
Fat. 2: Surfactant. 3  0  0  0  0.0000  0.3333 -0.3333 -0.3333  0 -1  0 -1
Fat. 3: Surfactant. 3  0  0  0 -0.3333 -0.3333 -0.3333 -0.3333  1  1  1  1

```

Here we see one of the appealing properties of Type III sum of squares: the hypothesis tested by the Type III sum of squares for Flour only involves parameters of the Flour term, whereas the hypothesis tested by the Type I sum of squares for Flour involves the parameters of Fat, Surfactant and Fat  $\times$  Surfactant.

The marginal means can also be obtained from `multicomp` using `comparisons="none"`. Doing so, we obtain the estimable functions for the marginal means for the over-parameterized model. For example, the estimable functions for the Fat marginal means are:

```

> Fat.mcomp<-multicomp(Baking.aov, focus="Fat", comp="none")

```

```
> round(Fat.mcomp$lm, 4)
              1      2      3
(Intercept) 1.0000 1.0000 1.0000
Flour. 1    0.2500 0.2500 0.2500
Flour. 2    0.2500 0.2500 0.2500
Flour. 3    0.2500 0.2500 0.2500
Flour. 4    0.2500 0.2500 0.2500
Fat. 1      1.0000 0.0000 0.0000
Fat. 2      0.0000 1.0000 0.0000
Fat. 3      0.0000 0.0000 1.0000
Surfactant. 1 0.3333 0.3333 0.3333
Surfactant. 2 0.3333 0.3333 0.3333
Surfactant. 3 0.3333 0.3333 0.3333
Fat. 1: Surfactant. 1 0.3333 0.0000 0.0000
Fat. 2: Surfactant. 1 0.0000 0.3333 0.0000
Fat. 3: Surfactant. 1 0.0000 0.0000 0.3333
Fat. 1: Surfactant. 2 0.3333 0.0000 0.0000
Fat. 2: Surfactant. 2 0.0000 0.3333 0.0000
Fat. 3: Surfactant. 2 0.0000 0.0000 0.3333
Fat. 1: Surfactant. 3 0.3333 0.0000 0.0000
Fat. 2: Surfactant. 3 0.0000 0.3333 0.0000
Fat. 3: Surfactant. 3 0.0000 0.0000 0.3333
```

The reader can verify that the Type III estimable functions for Fat are the differences between columns 1 and 3, and between columns 2 and 3.

### Sigma Constrained Parameterization

The function `lm` reparameterizes the linear model in an attempt to make the model matrix full column rank. We will next explore the computation of the adjusted means and the Type III sum of squares for Fat using the sigma constrained linear model. The sigma constraints were used in the `aov` fit above (`aov` calls `lm` with `singular.ok=T`). This was done by specifying `contr.sum` in the `contrasts` argument. In this setting the adjusted means can be computed with the following estimable functions:

---

```
> L
```

```

              Fat. 1 Fat. 2 Fat. 3
(Intercept)      1      1      1
  Flour1          0      0      0
  Flour2          0      0      0
  Flour3          0      0      0
    Fat1          1      0     -1
    Fat2          0      1     -1
Surfactant1       0      0      0
Surfactant2       0      0      0
Fat1Surfactant1   0      0      0
Fat2Surfactant1   0      0      0
Fat1Surfactant2   0      0      0
Fat2Surfactant2   0      0      0

```

Some justification to these functions may be in order: The parameterization chosen constrains the sum of the level estimates of each effect to zero. That is,

$$\sum_i b_i = \sum_j f_j = \sum_k s_k = \sum_j (fs)_{jk} = \sum_k (fs)_{jk} = 0.$$

Therefore, any effect that we are summing over in the mean estimate vanishes. The intercept in the least squares fit estimates  $\mu$  and the two coefficients for the Fat effect (labeled in L as Fat1 and Fat2) estimate  $f1$  and  $f2$ , respectively and  $f3 = -f1 - f2$ .

We can check that each function is, in fact, estimable by ensuring that they are in the row space of X, then compute the adjusted means.

```

> X<-model.matrix(Baking.aov)
> ls.f<-lsfit(t(X)%*%X, L, intercept=F)
> apply(abs(ls.f$residuals), 2, max)<0.0001
      Fat. 1 Fat. 2 Fat. 3
         T      T      T
> m<-t(L)%*%Baking.aov$coefficients

```

```
> m
      [, 1]
Fat. 1 5.850197
Fat. 2 6.577131
Fat. 3 7.472514
```

Now use the summary method for the `lm` object to obtain  $(X^tX)^{-1}$  and  $\hat{\sigma}$  and compute the standard errors of the least squares means.

```
> Baki.ng.sum<-summary.lm(Baki.ng.aov)
> Baki.ng.sum$sigma*sqrt(diag(t(L)%*%
+ Baki.ng.sum$cov.unscaled)%%L))
[1] 0.1364894 0.1477127 0.1564843
```

A set of Type III estimable functions for Fat can be obtained using the contrasts generated by `contr.helmert`.

```
> contr.helmert(3)
      [, 1] [, 2]
1      -1    -1
2       1    -1
3       0     2
```

We will use this set of orthogonal contrasts to test  $\bar{\mu}_{.1} = \bar{\mu}_{.2}$  and  $\bar{\mu}_{.1} + \bar{\mu}_{.2} = 2\bar{\mu}_{.3}$  which is equivalent to  $H_{\text{Fat}}$ .

```
> L.typeIII<-L%%contr.helmert(3)
> L.typeIII
```

```
      [, 1] [, 2]
(Intercept)    0    0
Flour1         0    0
Flour2         0    0
Flour3         0    0
Fat1          -1   -3
Fat2           1   -3
Surfactant1    0    0
Surfactant2    0    0
Fat1Surfactant1 0    0
Fat2Surfactant1 0    0
Fat1Surfactant2 0    0
Fat2Surfactant2 0    0
```

---

Finally, the Type III sum of squares is computed for Fat.

```
> h.m<-t(contr.helmert(3))%*%m
> t(h.m)%*%solve(t(L.typeIII)%*%Baki ng. sum$cov.unscaled%*%
+ L.typeIII)%*%h.m

      [, 1]
[1, ] 10.11785
```

Since we used the sigma-constrained model and the data is complete, we can also use drop1 to obtain the Type III sum of squares.

```
> drop1(Baki ng. aov, ~. )
```

Single term deletions

Model :

Speci fic. Vol	~	Fl our	+	Fat	*	Surfactant			
		Df	Sum of Sq			RSS	F Value		Pr(F)
<none>						2.31586			
Fl our	3	8.69081				11.00667	17.51280	0.00005181	
Fat	2	10.11785				12.43371	30.58263	0.00000778	
Surfactant	2	0.99721				3.31307	3.01421	0.08153989	
Fat: Surfactant	4	5.63876				7.95462	8.52198	0.00105692	

For the sigma-constrained model, the hypotheses  $H_{\text{Fat}}$  and  $H_{\text{surfactant}}$  can also be expressed as

$$H^*_{\text{Fat}}: f_1 = f_2 = 0$$

$$H^*_{\text{Surfactant}}: s_1 = s_2 = s_3 = 0.$$

The row for Fat in the drop1 ANOVA table is the reduction in sum of squares due to Fat given all other terms are in the model. This simultaneously tests that the least squares coefficients  $\beta_{\text{Fat}1} = f1$  and  $\beta_{\text{Fat}2} = f2$  are zero (and, hence  $f3 = -(f1 + f2)$  is zero) (Searle, 1987). The same argument applies to Surfactant. It follows that the following Type III estimable functions for Fat can be used to test  $H^*_{\text{Fat}}$  (or equivalently  $H_{\text{Fat}}$ ).

```
> L. typeIII
              [, 1] [, 2]
(Intercept)    0    0
  Flour1       0    0
  Flour2       0    0
  Flour3       0    0
   Fat1        1    0
   Fat2        0    1
Surfactant1     0    0
Surfactant2     0    0
Fat1Surfactant1 0    0
Fat2Surfactant1 0    0
Fat1Surfactant2 0    0
Fat2Surfactant2 0    0

> h. c<-t(L. typeIII)%*%Baking. aov$coef
> t(h. c)%*%solve(t(L. typeIII)%*%Baking. sum$cov. unscal ed)%*%
+ L. typeIII)%*%h. c
              [, 1]
[1, ] 10.11785
```

## References

- Milliken, G. A., Johnson, D. E., *Analysis of Messy Data Volume I: Designed Experiments*, Van Nostrand Reinhold Co., 473 pp.
- SAS Institute, Inc. (1990) *SAS/Stat User's Guide*, Fourth Edition. SAS Institute, Inc., pp 120-121

# POWER AND SAMPLE SIZE

# 11

---

<b>Normal Power And Sample Size</b>	<b>124</b>
Model Page	125
Options Page	126
<b>Binomial Power And Sample Size</b>	<b>130</b>
Model Page	131
Options Page	131
Printout Page	133
<b>Power and Sample Size Theory</b>	<b>135</b>
<b>Normally Distributed Data</b>	<b>136</b>
One-Sample Test of Gaussian Mean	136
Comparing Means From Two Samples	139
<b>Binomial Data</b>	<b>142</b>
References	148

When contemplating a study, one of the first statistical questions that arises is ‘How big does my sample need to be?’ The required sample size is a function of the alternative hypothesis, the probabilities of Type I and Type II errors, and the variability of the population(s) under study. Two new functions are available for computing power and sample size requirements, `normal.sample.size` and `binomial.sample.size`. Depending on the input, these functions will provide:

- For given power and alternative hypothesis, the required sample size
- For given sample size and power, the detectable difference
- For given sample size and alternative hypothesis, the power to distinguish between the hypotheses

These functions can be applied in one and two-sample studies, and will produce a table from vectorized input suitable for passing to Trellis graphics.

## NORMAL POWER AND SAMPLE SIZE

The Normal Power and Sample Size dialog assists in computing power, sample size or minimum detectable difference. Choose **Statistics:Power and Sample Size:Normal Mean** from the main menu. The dialog shown below appears.

The screenshot shows the 'Normal Power and Sample Size' dialog box with the 'Model' tab selected. The dialog is organized into several sections: 'Compute' with radio buttons for 'Sample Size' (selected), 'Power', and 'Min. Difference'; 'Sample Type' set to 'One Sample'; 'Probabilities' with 'Alpha(s)' at 0.05 and 'Power(s)' at 0.8; 'Sample Sizes' with empty fields for N1, N2, and N2 / N1; 'Standard Deviations' with 'Sigma1' at 1 and empty fields for Sigma2 and Sigma(X2 - X1); 'Null Hypothesis' with 'Mean' at 0 and an empty field for Mean1; 'Alternative Hypothesis' with an empty field for 'Alt Mean', an empty field for 'Mean2', and 'Test Type' set to 'two.sided'; and 'Results' with an empty field for 'Save As' and a checked 'Print Results' checkbox. At the bottom are buttons for 'OK', 'Cancel', 'Apply', '< >' (navigating to the 'current' tab), and 'Help'.

Normal Power and Sample Size	
Model Options Printout	
Select	
Compute	<input checked="" type="radio"/> Sample Size <input type="radio"/> Power <input type="radio"/> Min. Difference
Sample Type	One Sample
Probabilities	
Alpha(s)	0.05
Power(s)	0.8
Sample Sizes	
N1	
N2	
N2 / N1	
Standard Deviations	
Sigma1	1
Sigma2	
Sigma(X2 - X1)	
Null Hypothesis	
Mean	0
Mean1	
Alternative Hypothesis	
Alt Mean	
Mean2	
Test Type	two.sided
Results	
Save As	
<input checked="" type="checkbox"/> Print Results	
OK Cancel Apply < > current Help	

Figure 11.1: *The Normal Power and Sample Size dialog, Model page.*



## Model Page

### Select Group

#### Compute

Choose one of 'Sample Size' (default), 'Power' or 'Min. Difference'.

#### Sample Type

The choices are 'One Sample', 'Two Sample' or 'Paired'.

### Probabilities Group

This group is where alpha and power are specified, defined as

$\alpha = \Pr(\text{reject Null hypothesis if true})$

$\text{power} = \Pr(\text{reject Null hypothesis if false})$

You can select multiple values using the CTRL key, or you can type in values separated by commas.

### Sample Sizes Group

If computing power or minimum difference, samples sizes are input here. For two-sample tests, any two of N1, N2, N2/N1 will designate the third. In most cases it is natural to think in terms of N1 and N2/N1.

### Standard Deviations Group

For a one-sample test, 'Sigma1' is required. For a paired test, the standard deviation of the difference between samples is required, so the 'Sigma(X2 - X1)' field becomes active in place of 'Sigma1'. For a two-sample test, 'Sigma2' defaults to 'Sigma1'. Multiple values for the standard deviations can be input, separated by commas.

### Null Hypothesis Group

For a one-sample test, the mean is required, with a default value of 0. For a two-sample test, 'Mean1' is asked for.

### Alternative Hypothesis Group

For a one-sample test, the alternative mean is needed; for a two-sample test, 'Mean2' is requested.

#### Test Type

If the alternative hypothesis is one of inequality, the test type is 'two.sided'. Other choices are 'greater' and 'less'.

### Results Group

#### Save As

To save the resulting table as an S-PLUS object, type the name for the object here.

### Print Results

If this box is checked, the output will be printed to the Report window.

**Options Page**    The Options page is shown below.



**Figure 11.2:** *The Normal Power and Sample Size dialog, Options page.*

<b>Recompute Power</b>	By default, sample sizes are rounded up to the next integer value. Checking this option causes the power to be recomputed for the rounded sample size value.
<b>Exact N</b>	Checking this results in the exact value of N being returned, with no rounding.
<b>Interactive</b>	With this option checked, the results of the computations are written back to the dialog.
<b>Expand Input</b>	This causes the input to be expanded into a table where all combinations of input are used. For example, if you input two different powers and three alternative means, the resulting table will have six rows. If this option is unchecked, the above example will produce a table with three rows.
<b>Printout Page</b>	The 'Printout' page looks like this:

Columns	Digits
Mean1 (or Null) 1	Mean1 (or Null) 7
Sigma1 (or Paired) 2	Sigma1 (or Paired) 7
Mean2 (or Alt) 3	Mean2 (or Alt) 7
Sigma2 omit	Sigma2 0
Delta 4	Delta 7
Alpha 5	Alpha 7
Power 6	Power 7
N1 7	N1 7
N2 omit	N2 0
N2/N1 omit	N2/N1 0

Reset Fill Down

Save Object Object Name

Export Object Text File

OK Cancel Apply < > current Help

**Figure 11.3:** *The Normal Power and Sample Size dialog, Printout page.*

### Columns Group

This group allows you to control which columns are printed and in what order. To drop a column, choose 'omit'. When a column number is changed, the others are adjusted accordingly. For example, in the above dialog if you were to change 'Alpha' to 7, 'Power' and 'N1' would each be reduced by 1. Pressing the 'Reset' button will restore the values to their defaults.

### Digits Group

The number of digits can be controlled for each column individually. Pressing the 'Fill Down' button will copy the last selected digit down the list.

**Save Object  
Group****Object Name**

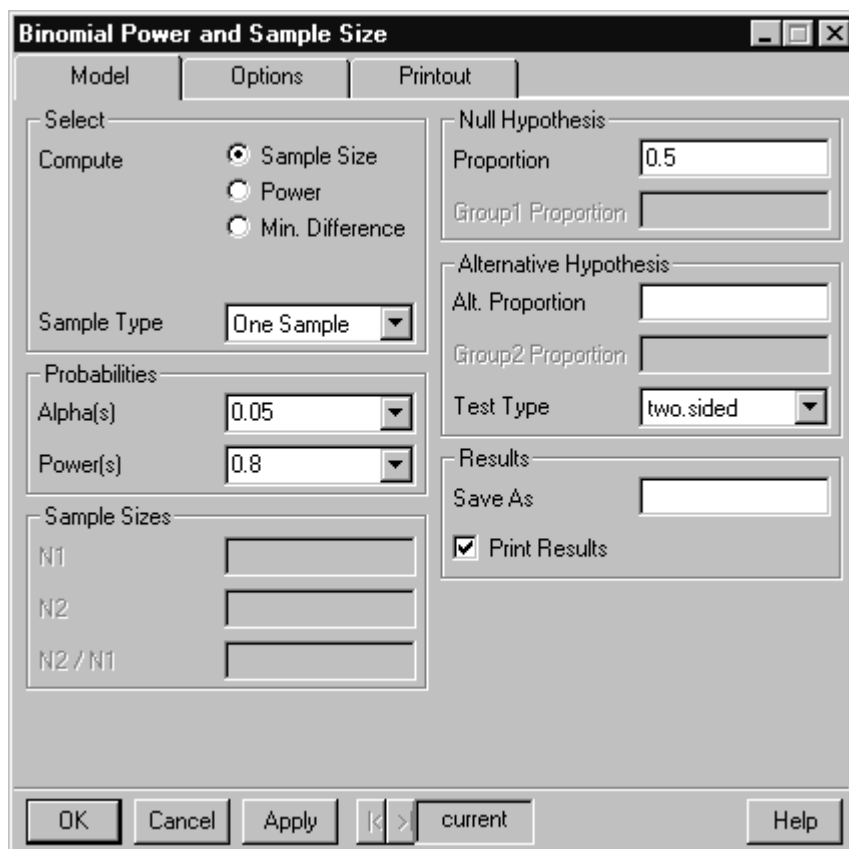
If you enter a name here, the printed table will be saved as a data.frame in the working directory.

**Export Object  
Group****Text File**

Entering a file name (or full path) will produce a tab-delimited text file. For more complete exporting capabilities, save the table as a data.frame (see above) and then choose **File:Export Data** from the menu.

## BINOMIAL POWER AND SAMPLE SIZE

The Binomial Power and Sample dialog assists in computing power, sample size or minimum detectable difference. Choose **Statistics:Power and Sample Size:Binomial Proportion** from the main menu. The dialog shown below appears.



**Figure 11.4:** *The Binomial Power and Sample Size dialog, Model page.*

## Model Page

### Select Group

#### Compute

Choose one of 'Sample Size' (default), 'Power' or 'Min. Difference'.

#### Sample Type

The choices are 'One Sample' or 'Two Sample'.

### Probabilities Group

This group is where alpha and power are specified, defined as

$\alpha = \Pr(\text{reject Null hypothesis if true})$

$\text{power} = \Pr(\text{reject Null hypothesis if false})$

You can select multiple values using the CTRL key, or you can type in values separated by commas.

### Sample Sizes Group

If computing power or minimum difference, samples sizes are input here. For two-sample tests, any two of N1, N2, N2/N1 will designate the third. In most cases it is natural to think in terms of N1 and N2/N1.

### Null Hypothesis Group

For a one-sample test, the proportion is required, with a default value of 0.50. For a two-sample test, 'Group1 Proportion' is asked for.

### Alternative Hypothesis Group

For a one-sample test, the alternative proportion is needed; for a two-sample test, 'Group2 Proportion' is requested.

#### Test Type

If the alternative hypothesis is one of inequality, the test type is 'two.sided'. Other choices are 'greater' and 'less'.

### Results Group

#### Save As

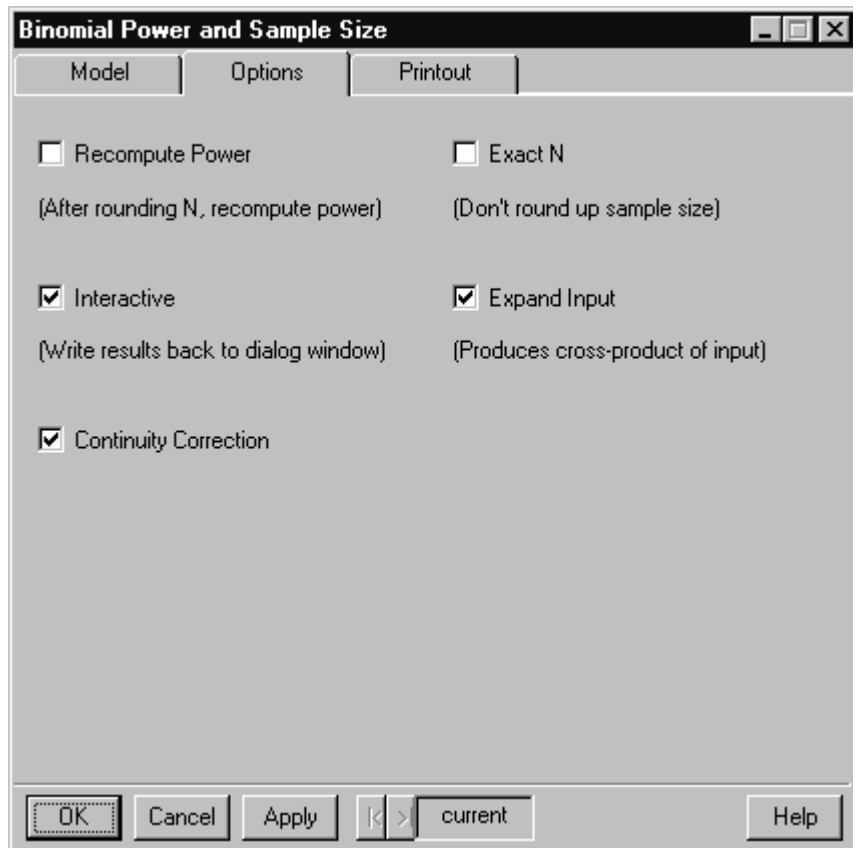
To save the resulting table as an S-PLUS object, type the name for the object here.

#### Print Results

If this box is checked, the output will be printed to the Report window.

## Options Page

The Options page is shown below.



**Figure 11.5:** *The Binomial Power and Sample Size dialog, Options page.*

<b>Recompute Power</b>	By default, sample sizes are rounded up to the next integer value. Checking this option causes the power to be recomputed for the rounded sample size value.
<b>Exact N</b>	Checking this results in the exact value of N being returned, with no rounding.
<b>Interactive</b>	With this option checked, the results of the computations are written back to the dialog.
<b>Expand Input</b>	This causes the input to be expanded into a table where all combinations of input are used. For example, if you input two different powers and three



alternative means, the resulting table will have six rows. If this option is unchecked, the above example will produce a table with three rows.

### Continuity Correction

With this option checked, a continuity correction is used in the computations.

**Printout Page** The 'Printout' page looks like this:

**Binomial Power and Sample Size**

Model Options **Printout**

**Columns**

Prop1 (or Null) 1

Prop2 (or Alt) 2

Delta 3

Alpha 4

Power 5

N1 6

N2 omit

N2/N1 omit

Reset

**Digits**

Prop1 (or Null) 7

Prop2 (or Alt) 7

Delta 7

Alpha 7

Power 7

N1 7

N2 0

N2/N1 0

Fill Down

Save Object  
Object Name

Export Object  
Text File

OK Cancel Apply < > current Help

**Figure 11.6:** *The Binomial Power and Sample Size dialog, Printout page.*

### Columns Group

This group allows you to control which columns are printed and in what order. To drop a column, choose 'omit'. When a column number is changed,

the others are adjusted accordingly. For example, in the above dialog if you were to change 'Alpha' to 7, 'Power' and 'N1' would each be reduced by 1. Pressing the 'Reset' button will restore the values to their defaults.

**Digits Group**

The number of digits can be controlled for each column individually. Pressing the 'Fill Down' button will copy the last selected digit down the list.

**Save Object Group****Object Name**

If you enter a name here, the printed table will be saved as a data.frame in the working directory.

**Export Object Group****Text File**

Entering a file name (or full path) will produce a tab-delimited text file. For more complete exporting capabilities, save the table as a data.frame (see above) and then choose **File:Export Data** from the menu.

# POWER AND SAMPLE SIZE THEORY

When designing a study, one of the first questions to arise is “How large does my sample size need to be?” Intuitively, we have a sense that this depends on how small a difference we're trying to detect, how much variability is inherent in our data, and how certain we want to be of our results. In a classical hypothesis test of  $H_o$  (null hypothesis) versus  $H_a$  (alternative hypothesis), there are four possible outcomes, two of which are erroneous:

- Don't reject  $H_o$  when  $H_o$  is true.
- Reject  $H_o$  when  $H_o$  is false.
- Reject  $H_o$  when  $H_o$  is true (type I error).
- Don't reject  $H_o$  when  $H_o$  is false (type II error).

To construct a test, the distribution of the test statistic under  $H_o$  is used to find a critical region which will ensure the probability of committing a type I error does not exceed some predetermined level. This probability is typically denoted  $\alpha$ . The *power* of the test is its ability to *correctly reject* the null hypothesis, or  $1 - \text{Pr}(\text{type II error})$ , which is based on the distribution of the test statistic under  $H_a$ . The required sample size then will be a function of

1. The null and alternative hypotheses.
2. The target  $\alpha$ .
3. The desired power to detect  $H_a$ .
4. The variability within the population(s) under study.

Our objective is, for a given test, to find a relationship between the above factors and the sample size that will enable us to select a sample size consistent with the desired  $\alpha$  and power.

## NORMALLY DISTRIBUTED DATA

### One-Sample Test of Gaussian Mean

When conducting a one-sample test of a normal mean, we start by writing our assumptions and hypotheses:

$$X_i \sim N(\mu, \sigma^2)$$

where  $i = 1, \dots, n$ , and  $\sigma^2$  is known. To perform a two-sided test of equality the hypotheses would be as follows:

$$H_o: \mu = \mu_o$$

$$H_a: \mu = \mu_a$$

Our best estimate of  $\mu$  is the sample mean, which is normally distributed:

$$\bar{X} \sim N\left(\mu, \frac{\sigma^2}{n}\right)$$

and the test statistic is

$$Z = \sqrt{n}(\bar{X} - \mu_o) / \sigma$$

$$\sim N(\mu - \mu_o, 1)$$

$$\sim N(0, 1) \text{ for } H_o$$

Reject  $H_o$  if  $|Z| > Z_{(1-\alpha/2)}$ , which guarantees a level  $\alpha$  test. The power of the test to detect  $\mu = \mu_a$  is

$$\begin{aligned} \text{Power} = & \Phi\left(\frac{\sqrt{n}(\mu_o - \mu_a)}{\sigma} - Z_{1-\alpha/2}\right) \\ & + \Phi\left(\frac{\sqrt{n}(\mu_a - \mu_o)}{\sigma} - Z_{1-\alpha/2}\right) \end{aligned}$$

We can think of the left side of the sum as the *lower power*, or the power to detect  $\mu_a < \mu_o$ , and the right side as the *upper power*, or the power to detect  $\mu_a > \mu_o$ . Solving for  $n$  using both upper and lower power would be difficult, but we note that when  $\mu_a - \mu_o < 0$ , the upper power is negligible ( $< \alpha/2$ ) and similarly the lower power is small when  $\mu_a - \mu_o > 0$ . So the equation can be simplified by using the absolute value of the difference between  $\mu_a$  and  $\mu_o$  and considering only one side of the sum. This results in the following sample size formula:

$$n = [(\sigma(Z_{1-\alpha/2} + Z_{Power})) / |\mu_a - \mu_o|]^2$$

### Comments

- While only one of upper power and lower power is used in deriving the sample size formula, the S-PLUS functions for computing power and sample size uses both the upper and lower power when computing the *power* of a two-tailed test for a given sample size.
- In practice, the variance of the population is seldom known and the test statistic is based on the *t-distribution*. Using the t-distribution to derive sample size requires an iterative approach, since the sample size is needed to specify the degrees of freedom. The difference between the quantile value for the t-distribution versus the standard normal is only significant when small sample sizes are required, so the standard formula based on the normal distribution was chosen. Keep in mind that for samples sizes less than 10, the power of a t-test could be significantly less than the target power.
- The formula for a one-tailed test is derived along similar lines, and is exactly the same as the two-tailed formula with the exception that  $Z_{(1-\alpha/2)}$  is replaced by  $Z_{(1-\alpha)}$ .

### Examples

The function for computing sample size for normally distributed data is `normal.sample.size`. This function can be used to compute sample size,

power, or minimum detectable difference and will automatically chose what to compute based on what information is input. Here are some simple examples:

```
#
# one-sample case, using all the defaults
#
> normal.sample.size(mean.alt = 0.3)
  mean.null sd1 mean.alt del.ta alpha power n1
1         0   1      0.3   0.3  0.05  0.888

#
# reduce output with summary
#
> summary(normal.sample.size(mean.alt = 0.3))
del.ta power n1
1   0.3   0.888

#
# upper-tail test recomputing power
#
> normal.sample.size(mean = 100, mean.alt = 105, sd1 = 10,
  power = c(.8, .9, .95, .99), alt = "greater",
  recompute.power = T)
  mean.null sd1 mean.alt del.ta alpha      power n1
1       100  10      105     5  0.05 0.8037649 25
2       100  10      105     5  0.05 0.9054399 35
3       100  10      105     5  0.05 0.9527153 44
4       100  10      105     5  0.05 0.9907423 64

#
# calculate power
#
> normal.sample.size(mean = 100, mean.alt = 105, sd1 = 10,
  n1 = (1:5)*20)
  mean.null sd1 mean.alt del.ta alpha      power n1
1       100  10      105     5  0.05 0.6087795 20
2       100  10      105     5  0.05 0.8853791 40
3       100  10      105     5  0.05 0.9721272 60
4       100  10      105     5  0.05 0.9940005 80
5       100  10      105     5  0.05 0.9988173 100
```

```
#
# lower-tail test, minimum detectable difference
#
> summary(normal.sample.size(mean = 100, sd1 = 10, n1 =
(1:5)*20,
  power = .9, alt = "l"))
  mean.alt      delta power  n1
1 93.45636 -6.543641    0.9  20
2 95.37295 -4.627053    0.9  40
3 96.22203 -3.777973    0.9  60
4 96.72818 -3.271821    0.9  80
5 97.07359 -2.926405    0.9 100
```

See the online help files for `normal.sample.size` and `summary.power.table` for more details.

## Comparing Means From Two Samples

Extending this formula to two-sampled tests, is relatively easy. Given two independent samples from normal distributions

$$X_{1,i} \sim N(\mu_1, \sigma_1^2) \quad i = 1, \dots, n_1$$

$$X_{2,j} \sim N(\mu_2, \sigma_2^2) \quad j = 1, \dots, n_2$$

where  $n_2 = kn_1$ , we'll construct a two-sided test of equality of means

$$H_o: \mu_1 = \mu_2$$

$$H_a: \mu_1 \neq \mu_2$$

which is more conveniently written

$$H_o: \mu_2 - \mu_1 = 0$$

$$H_a: \mu_2 - \mu_1 \neq 0$$

The difference of the sample means is normally distributed

$$\begin{aligned}(\bar{X}_2 - \bar{X}_1) &\sim N\left(\mu_2 - \mu_1, \frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}\right) \\ &\sim N\left(\mu_2 - \mu_1, \frac{1}{n_1}\left(\sigma_1^2 + \frac{\sigma_2^2}{k}\right)\right)\end{aligned}$$

which leads to the test statistic

$$Z = \frac{\bar{X}_2 - \bar{X}_1}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

Derivation of the two-sample formulas proceed along the same lines as the one-sample case, producing the following formulas:

$$\begin{aligned}n_1 &= \left(\sigma_1 + \frac{\sigma_2^2}{k}\right) \left[ \frac{(Z_{(1-\alpha/2)} + Z_{Power})}{|\mu_2 - \mu_1|} \right]^2 \\ n_2 &= kn_1\end{aligned}$$

### Examples:}

For two-sample cases, use `normal.sample.size` with `mean2` instead of `mean.alt`:

```
#
# Don't round sample size
#
> summary(normal.sample.size(mean2 = 0.3, exact.n = T))
del ta power      n1      n2
1    0.3    0.8 174.4195 174.4195

#
# round sample size, then recompute power
#
> summary(normal.sample.size(mean2 = 0.3, recompute = T))
del ta      power  n1  n2
```



---

```
1 0.3 0.8013024 175 175

#
# Unequal sample sizes, lower tail test
#
> normal.sample.size(mean = 100, mean2 = 94, sd1 = 15,
prop.n2 = 2,
power = 0.9, alt = "less")
mean1 sd1 mean2 sd2 del ta alpha power n1 n2 prop.n2
1 100 15 94 15 -6 0.05 0.981 162 2
```

## BINOMIAL DATA

### One-Sample Test of Binomial Proportion

Another very common test is for a *binomial proportion*. Say we have data sampled from a binomial distribution,

$$X_i \sim B(\pi, n), i = 1, \dots, n$$

Each  $X_i$  represents the number of 'successes' observed in  $n$  *Bernoulli trials*, where  $\Pr(\text{success}) = \pi$ . The mean and variance of the random variable  $X$  is

$$\begin{aligned} E(X) &= n\pi \\ \text{Var}(X) &= n\pi(1 - \pi) \end{aligned}$$

We wish to test the value of the parameter  $\pi$ , using a two-sided test.

$$H_o: \pi = \pi_o$$

$$H_a: \pi = \pi_a$$

We could use an exact binomial test, but for sufficiently large  $n$ , and if the distribution is not too skewed, ( $\pi$  is not too close to 0 or 1), a normal approximation can be used. A good rule of thumb is that the normal distribution will be a good approximation to the binomial distribution if

$$n\pi(1 - \pi) \geq 5$$

When using a continuous distribution to approximate a discrete one, a *continuity correction* is usually recommended; typically, a value of 1/2 is used

to extend the range in either direction, so

$$Pr(X_l \leq X \leq X_u)$$

using a binomial distribution, becomes

$$Pr\left(X_l - \frac{1}{2} \leq X \leq X_u + \frac{1}{2}\right)$$

when using a normal approximation. If the continuity correction is temporarily suppressed, the sample size formula is derived very much as in the normal case:

$$n^* = \left[ \frac{\sqrt{\pi_o(1-\pi_o)}Z_{(1-\alpha/2)} + \sqrt{\pi_o(1-\pi_o)}Z_{Power}}{|\pi_a - \pi_o|} \right]^2$$

There have been several suggestions concerning how to best incorporate a continuity correction into the sample-size formula. The one adopted in the S-PLUS function `binomial.sample.size` for a one-sample test is

$$n = n^* + \frac{2}{|\pi_a - \pi_o|}$$

## Examples

```
#
# one-sample case, using all the defaults
#
> binomial.sample.size(p.alt = 0.3)
      p.null  p.alt delta alpha power n1
1      0.5    0.3  -0.2  0.05   0.8 37
```

```
#
# minimal output
#
> summary(binomial.sample.size(p.alt = 0.3))
      delta power n1
1 -0.2    0.8 37

#
# compute power
#
binomial.sample.size(p = .2, p.alt = .12, n1 = 250)
      p.null p.alt delta alpha      power  n1
1    0.2    0.12 -0.08  0.05 0.8997619 250
```

### Comparing Proportions From Two Samples

The two-sample test for proportions is a bit more involved than the others we've looked at. Say we have data sampled from two binomial distributions

$$\begin{aligned} X_{1,i} &\sim B(\pi_1, n_1), & i &= 1, \dots, n_1 \\ X_{2,j} &\sim B(\pi_2, n_2), & j &= 1, \dots, n_2 \end{aligned}$$

where  $n_2 = kn_1$ , we'll construct a two-sided test of equality of means

$$H_o: \pi_1 = \pi_2$$

$$H_a: \pi_1 \neq \pi_2$$

which is more conveniently written

$$H_o: \pi_1 - \pi_2 = 0$$

$$H_a: \pi_1 - \pi_2 \neq 0$$

Using our best estimator of the parameter  $\pi$ , we can begin constructing a test

statistic:

$$\hat{\pi}_1 = \frac{1}{n_1} \sum_{i=1}^{n_1} X_{1,i}$$

$$\hat{\pi}_2 = \frac{1}{n_2} \sum_{j=1}^{n_2} X_{2,j}$$

$$\hat{\pi}_2 - \hat{\pi}_1 \sim N\left(\pi_2 - \pi_1, \frac{\pi_1(1 - \pi_1)}{n_1} + \frac{\pi_2(1 - \pi_2)}{n_2}\right)$$

$$\hat{\pi}_2 - \hat{\pi}_1 \sim N\left(\pi_2 - \pi_1, \frac{1}{n_1} \left( \pi_1(1 - \pi_1) + \frac{\pi_2(1 - \pi_2)}{k} \right)\right)$$

In the case where the null hypothesis is true, so  $\pi_2 = \pi_1 = \pi$ , this can be written as

$$\hat{\pi}_2 - \hat{\pi}_1 \sim N\left(0, \frac{\pi(1 - \pi)}{n_1} \left(1 + \frac{1}{k}\right)\right)$$

Immediately a problem arises, namely, the variance needed to construct the test statistic depends on the parameters being tested. It seems reasonable to use all of the data available to estimate the variances, and that is exactly what is done. A weighted average of the two estimates for the proportions is used

to estimate the variance under  $H_o$ . The test statistic then is

$$\bar{\pi} = \frac{n_1 \hat{\pi}_1 + n_2 \hat{\pi}_2}{n_1 + n_2} = \frac{\hat{\pi}_1 + k \hat{\pi}_2}{1 + k}$$

$$Z = \frac{\hat{\pi}_2 - \hat{\pi}_1}{\sqrt{\bar{\pi}(1 - \bar{\pi})\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}$$

If the null hypothesis is true, this gives  $Z \sim N(0, 1)$ . We use this to derive the formula without continuity correction:

$$n_1^* = \left[ \frac{\sqrt{\pi_1(1 - \pi_1) + \frac{\pi_2(1 - \pi_2)}{k}} Z_{power} + \sqrt{\bar{\pi}(1 - \bar{\pi})\left(1 + \frac{1}{k}\right)} Z_{1 - \alpha/2}}{|\pi_2 - \pi_1|} \right]^2$$

Applying the two-sample adjustment for a continuity correction produces the final results

$$n_1 = n_1^* + \frac{k + 1}{k|\pi_2 - \pi_1|}$$

$$n_2 = kn_1$$

## Examples

```
#
# for two-sample, use p2 instead of p.at t
#
> summary(binomial.sample.size(p2 = 0.3))
delta power  n1  n2
1  -0.2    0.8 103 103

#
# Don't round sample size and don't use continuity
# correction
#
```

---

```

> summary(binomial.sample.size(p2 = 0.3, exact.n = T,
correct = F))
      del ta power      n1      n2
1  -0.2   0.8 92.99884 92.99884

#
# round sample size, then recompute power
#
> summary(binomial.sample.size(p2 = 0.3, recompute = T))
      del ta      power  n1  n2
1  -0.2 0.8000056 103 103

#
# Unequal sample sizes, lower tail test
#
> binomial.sample.size(p = .1, p2 = .25, prop.n2 = 2, power
= 0.9, alt = "less")
      p1    p2 del ta alpha power n1  n2 prop.n2
1 0.1 0.25  0.15  0.05   0.9 92 184      2

#
# Compute minimum detectable difference (del ta) given
sample size
# and power.
#
> binomial.sample.size(p = .6, n1 = 500, prop.n2 = .5, power
= c(.8, .9, .95))
      p1      p2      del ta alpha power  n1  n2 prop.n2
1 0.6 0.7063127 0.1063127  0.05  0.80 500 250    0.5
2 0.6 0.7230069 0.1230069  0.05  0.90 500 250    0.5
3 0.6 0.7367932 0.1367932  0.05  0.95 500 250    0.5

#
# compute power
#
> binomial.sample.size(p = 0.3, p2 = seq(0.31, 0.35,
by=0.01),
      n1 = 1000, prop.n2 = 0.5)
      p1    p2 del ta alpha      power  n1  n2 prop.n2
1 0.3 0.31  0.01  0.05 0.06346465 1000 500    0.5
2 0.3 0.32  0.02  0.05 0.11442940 1000 500    0.5

```

3	0.3	0.33	0.03	0.05	0.20446778	1000	500	0.5
4	0.3	0.34	0.04	0.05	0.32982868	1000	500	0.5
5	0.3	0.35	0.05	0.05	0.47748335	1000	500	0.5

**References**

Rosner, Bernard (1990). *Fundamentals of Biostatistics* (Third Edition). PWS-Kent, Boston.

Fisher, Lloyd D. and Van Belle, Gerald (1993). *Biostatistics* Wiley, New York.

Fleiss, Joseph L. (1981). *Statistical Methods for Rates and Proportions*. Wiley, New York.



# ROBUST LINEAR REGRESSION

# 12

---

<b>OVERVIEW OF THE ROBUST REGRESSION METHOD</b>	<b>151</b>
Key Robustness Features of the Method	151
The Essence of the Method: a Special M-Estimate	151
Using the lmRobMM Function to Obtain a Robust Fit	152
Comparison of Least Squares and Robust Fits	153
Robust Model Selection	153
<b>COMPUTING LEAST SQUARES AND ROBUST FITS</b>	<b>154</b>
Computing a Least Squares Fit	154
Computing a Robust Fit	155
Least Squares vs. Robust Fitted Model Objects	156
<b>VISUALIZING AND SUMMARIZING THE ROBUST FIT</b>	<b>157</b>
Visualizing the Fit with the plot Function	157
Statistical Inference with the summary Function	159
<b>COMPARING LEAST SQUARES AND ROBUST FITS</b>	<b>162</b>
Creating a Comparison Object for LS and Robust Fits	162
Visualizing LS vs. Robust Fits	162
Statistical Inference for LS vs. Robust Fits	164
<b>ROBUST MODEL SELECTION</b>	<b>166</b>
Robust F and Wald Tests	166
Robust FPE Criterion	167
<b>CONTROLLING OPTIONS FOR ROBUST REGRESSION</b>	<b>169</b>
Efficiency at Gaussian Model	169
Alternative Loss Function	169
Confidence Level of Bias Test	171
Resampling Algorithms	173
Random Resampling Parameters	173
Genetic Algorithm Parameters	174
<b>THEORETICAL DETAILS</b>	<b>175</b>
Initial Estimate Details	175
Optimal and Bisquare Rho and Psi-Functions	176
The Efficient Bias Robust Estimate	177
Efficiency Control	177

Robust R-Squared	177
Robust Deviance	179
Robust F Test	179
Robust Wald Test	179
Robust FPE (RFPE)	179
Appendix	180
<b>ROBUST MM REGRESSION</b>	<b>182</b>
<b>BIBLIOGRAPHY</b>	<b>194</b>

# OVERVIEW OF THE ROBUST REGRESSION METHOD

This section provides you with an overview of the tools at your disposal for computing a modern robust linear regression model in S-PLUS, including robust inference for coefficients and robust model selection. You find out how to use the robust regression tools in detail in the sections that follow.

## Key Robustness Features of the Method

You will learn how to fit a linear model using a modern robust method that has the following general features:

- In data-oriented terms, the robust fit is minimally influenced by outliers in the independent variables space, in the response (dependent variable) space, or in both.
- In probability-oriented terms, the robust fit minimizes the maximum possible (large sample size) coefficients estimate bias due to a non-Gaussian contamination distribution model which generates outliers, subject to achieving a desired (large sample size) coefficient estimates efficiency when the data has a Gaussian distribution.
- The statistical inference produced by the fit is based on large sample size approximations for such quantities as standard errors and “t-statistics” of coefficients, R-squared values, etc.

For further information read the section Theoretical Details below.

## The Essence of the Method: a Special M-Estimate

You are fitting a general linear model of the form

$$y_i = x_i^T \beta + \varepsilon_i, \quad i = 1, \dots, n$$

with p-dimensional independent predictor (independent) variables  $x_i$  and coefficients  $\beta$ , and scalar response (dependent) variable  $y_i$ . S-PLUS 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  $\rho$  is a particular optimal symmetric *bounded* loss function, described in the *Theoretical Details* section. The shape of this optimal function is shown in Figure 4 below.

Alternatively  $\hat{\beta}$  is a solution of the estimating equation

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

where  $\psi = \rho'$  is a redescending (non-monotonic) function.

A key issue is that since  $\rho$  is bounded, it is non-convex, and the minimization above can have many local minima. Correspondingly, the estimating equation above can have multiple solutions. S-PLUS deals with this by computing highly robust initial estimates  $\hat{\beta}$  and  $\hat{s}$  with breakdown point 0.5, using the S-estimate approach described in the *Theoretical Details* section, and 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).<sup>1</sup>

S-PLUS also provides for an automatic choice between the initial and final estimates based on evaluating the potential bias of the final estimate.

## Using the `lmRobMM` Function to Obtain a Robust Fit

You will compute a robust regression fit using the `lmRobMM` function. The resulting robustly fitted model object is almost identical in structure to a least squares fitted model object returned by `lm`, i.e., you will get most of the same fitted model components, such as coefficient standard errors and t-statistics, etc.

- 
1. The theory for this new robust method is based on Rousseeuw and Yohai (1984), Yohai, Stahel, and Zamar (1991), and Yohai and Zamar (1998). The code is based on the ROBETH library of Alfio Marazzi, with additional work by R. Douglas Martin, Douglas B. Clarkson, and Jeffrey Wang of MathSoft, partially supported by an SBIR Phase I grant entitled "Usable Robust Methods" funded by the National Institutes of Health.

## **Comparison of Least Squares and Robust Fits**

In order to facilitate comparison of least squares and robust fits of a linear regression model, you use a special function to create an object with the relevant information from the least squares and robust fits, e.g., `t-statistics`, `residuals`, etc. You then use this object as arguments to the usual `S-PLUS` printing, summarizing and plotting functions to get tabular and graphical displays in a form that makes it easy for you to compare the results of the least squares and robust fits.

## **Robust Model Selection**

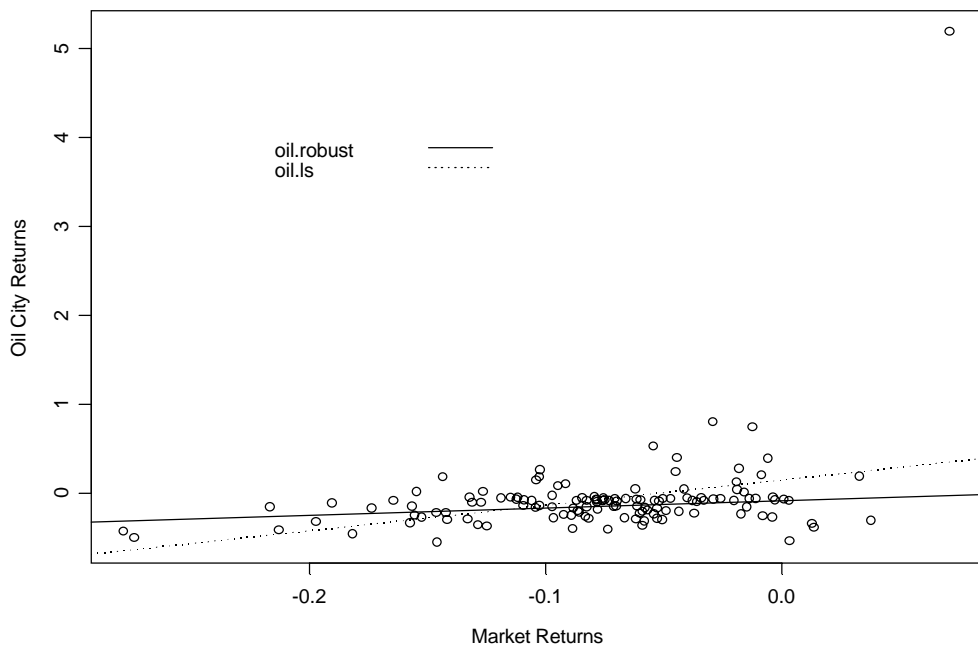
It is not enough for you to use a robust linear model fitting method when you are trying to decide which of several alternative models to use, based on alternative sets of predictor variables. You also need a robust model selection criterion. To this end, you may use one of the following three robust model selection criteria: robust F-test, robust Wald test, and robust FPE (RFPE) criterion.

## COMPUTING LEAST SQUARES AND ROBUST FITS

### Computing a Least Squares Fit

The S-PLUS data frame `oil.l.df` contains monthly excess returns on the stocks of Oil City Petroleum, Inc. from April 1979 to December 1989 and the monthly excess returns of the market of the same period. “Returns” are defined as the relative change in the price of the stock over a one-month interval, and “excess” means relative to the monthly return at the risk-free rate of a 90-day U.S. Treasury bill.

The scatter plot of the data is shown in Figure 1. Obviously there is one big outlier in the data.



**Figure 12.1:** *LS Fit and Robust Fit of `oil.l.df`*

Financial economists usually use LS to fit a straight line to a particular stock

return and the market return, and the estimated coefficient of the market return is called the “beta”, which measures the riskiness of the stock in terms of standard deviation and the expected returns. The larger the beta, the more risky the stock is compared with the market, but the larger the expected returns.

For comparison purposes, first fit an LS model to the data as follows:

```
> oil.ls <- lm(Oil ~ Market, data=oil.ls)
```

and print a short summary of the fitted model:

```
> oil.ls
Call:
lm(formula = Oil ~ Market, data = oil.df)

Coefficients:
(Intercept)  Market
  0.1474486  2.85674

Degrees of freedom: 129 total; 127 residual
Residual standard error: 0.4866656
```

## Computing a Robust Fit

To obtain a robust fit, you use the `lmRobMM` function just like the `lm` function:

```
> oil.robust <- lmRobMM(Oil ~ Market, data=oil.df)

> oil.robust
Final M-estimates.

Call:
lmRobMM(formula = Oil ~ Market, data = oil.df)

Coefficients:
(Intercept)  Market
 -0.08395777  0.8288791

Degrees of freedom: 129 total; 127 residual
Residual scale estimate: 0.1446283
```

Obviously, the robust estimate of beta is dramatically different from the LS estimate. According to the LS method, the beta of this stock is 2.857, which

implies that the stock is 2.857 times as volatile as the market, and has about 2.857 times the expected return. The robust estimate of beta is 0.829, which implies that the stock has somewhat less volatility and expected return than the market.

Also note that the robust scale estimate is 0.14, whereas the scale estimate from LS is 0.49. The LS scale estimate is based on the sum of squared residuals and thus considerably inflated by the presence of outliers in the data.

## Least Squares vs. Robust Fitted Model Objects

The object returned by the `lm` function for LS fit is of class `"lm"`:

```
> class(oi.l.l.s)
[1] "lm"
```

On the other hand, the object returned by `lmRobMM` is of class `"lmRobMM"`:

```
> class(oi.l.robust)
[1] "lmRobMM"
```

Just as with an object of class `"lm"`, you can easily visualize, print and summarize robust fit objects of class `"lmRobMM"` using the generic functions `plot`, `print` and `summary`.



# VISUALIZING AND SUMMARIZING THE ROBUST FIT

## Visualizing the Fit with the plot Function

For a simple linear regression, you can easily see outliers in the scatter plot, as in the above example. However, in multiple regression it is not so easy to tell if there are some outliers in the data, and what the outliers are. Nonetheless, S-PLUS makes it easy for you to visualize the outliers in a multiple regression. To illustrate this point, let us use the well known “stack loss” data set.

The S-PLUS product includes the stack loss data set which has been analyzed by a large number of statisticians. The stack loss in this data set is the percent loss (times 10) of ammonia during 21 days of operation. The ammonia is lost during the process of producing nitric acid by dissolving the ammonia in water. Three variables—air flow, water temperature, and acid concentration—may influence the loss of ammonia. The stack loss response data is contained in the vector `stack.loss`, and the three independent variables are contained in the matrix `stack.x`.

First, you combine the response and independent variables into a data frame `stack.df`:

```
> stack.df <- data.frame(Loss=stack.loss, stack.x)
```

Then you compute an LS fit object `stack.ls`:

```
> stack.ls <- lm(Loss ~ Air.Flow + Water.Temp + Acid.Conc.,
+               data =stack.df)
```

and finally compute a robust fit object `stack.robust`:

```
> stack.robust <- lmRobMM(Loss ~ Air.Flow + Water.Temp +
+                         Acid.Conc., data =stack.df)
```

Now you use the `plot` function to visualize the fit:

```
> plot(stack.robust)
```

Make a plot selection (or 0 to exit):

- 1: plot: All
- 2: plot: Residuals vs Fitted Values
- 3: plot: Sqrt of abs(Residuals) vs Fitted Values
- 4: plot: Response vs Fitted Values
- 5: plot: Normal QQplot of Residuals
- 6: plot: r-f spread plot

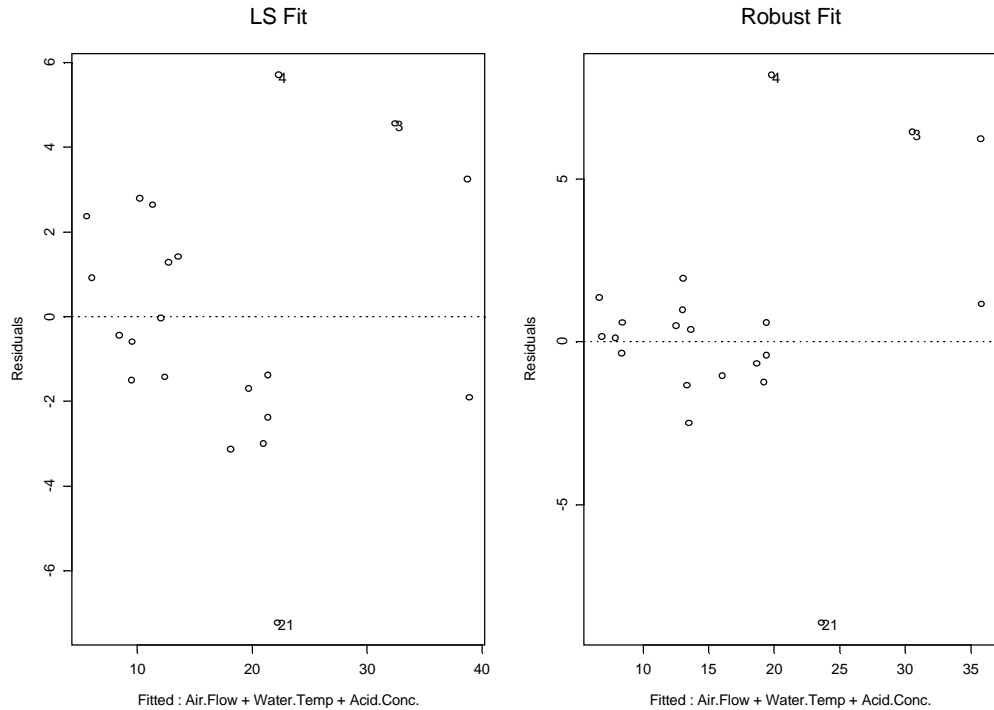
**Select i on:**

Note that Cook's distance is not currently available when a robust method is used.

Now you can compare the plot of residuals versus fitted values for both the LS fit and the robust fit using the following commands:

```
> par(mfrow=c(2, 1))  
  
> plot(stack.ls, which.plots=1)  
  
> plot(stack.robust, which.plots=1)
```

Figure 2 shows those two plots. As you can see, the robust fit pushes the outliers further away from the majority of the data, so that you can more easily identify the outliers.



**Figure 12.2:** *Residuals vs. Fitted Values: Stack Loss Data*

## Statistical Inference with the summary Function

The generic `summary` function provides you with the usual kinds of inference output, e.g., t-values and p-values along with some additive and useful information, including tests for bias. For example, to obtain more information about the robust fit `oi.l.robust`, use `summary` on this object:

```
> summary(oi.l.robust)
```

Final M-estimates.

```
Call: lmRobMM(formula = Oil ~ Market, data = oil.df)
```

```
Residuals:
```

```
      Min       1Q   Median       3Q      Max
-0.4566 -0.08875  0.03082  0.1031  0.218
```

```
Coefficients:
```

```
              Value Std. Error t value Pr(>|t|)
(Intercept) -0.0840   0.0281    -2.9929  0.0033
Market      0.8289   0.2834     2.9245  0.0041
```

```
Residual scale estimate: 0.1446 on 127 degrees of freedom
```

```
Proportion of variation in response explained by model:
0.05261
```

```
Test for Bias
```

```
              Statistics    P-value
M-estimate      2.16 0.3398475
LS-estimate     22.39 0.0000138
```

```
Correlation of Coefficients:
```

```
      (Intercept)
Market 0.8169
```

```
The seed parameter is : 1313
```

First note the standard errors, the t-values, and the p-values of the coefficients. The standard errors are computed from the robust covariance matrix of the estimates. For technical details about the computation of robust covariance matrix, refer to Yohai, Stahel and Zamar (1991).

Second, the summary method provides another piece of useful information: the “Proportion of variation in response explained by model”, usually known as  $R^2$ . S-PLUS calculates a robust version of  $R^2$ . The details of how the robust  $R^2$  is calculated can be found in the section *Theoretical Details of the Robust Regression Method*.

Finally, there is a “Test for Bias” section in the summary. This section provides the test statistics of the bias of the final M-estimates and the LS estimates against the initial S-estimates. In this case, the test for bias of the final M-estimates yields a p-value of 0.33, which suggests that the bias of the final M-estimates relative to the initial S-estimates is not significant at the usual level. That is why the “Final M-estimates” is reported in the first line of its summary output instead of the initial S-estimates. The test for bias of the

LS estimates relative to the S-estimates yields a p-value of 0, which indicates that the LS estimate is highly biased, so you strongly prefer to use the robust MM-estimator.

For technical details about how the tests for bias are calculated, see Yohai, Zamar and Stahel (1991).

## COMPARING LEAST SQUARES AND ROBUST FITS

### Creating a Comparison Object for LS and Robust Fits

In the section *Visualizing the Fit with the plot Function*, we compared the residuals vs. fitted values plot for both the LS and robust fits. You might have noted that the two plots do not have the same vertical scale. It would be nice to have the capability of plotting different fits on the same scale for easy visual comparison and also making tabular displays of LS and robust fits which are conveniently aligned for ease of comparing inference results. To this end S-PLUS provides a function `compare.fits`, for creating a models comparison object, along with appropriate `print`, `plot` and `summary` methods for this class of object.

For example, to compare the results from the two fits `oil.lm` and `oil.robust`, first create the comparison object `oil.cmpr` with the following command:

```
> oil.cmpr <- compare.fits(oil.lm, oil.robust)
```

The object returned by `compare.fits` is of class "compare.fits". Now you can print a short summary of the comparison:

```
> oil.cmpr
```

Call:

```
oil.lm lm(formula = Oil ~ Market, data = oil.df)
oil.robust lmRobMM(formula = Oil ~ Market, data = oil.df)
```

Coefficients:

	oil.lm	oil.robust
(Intercept)	0.1474	-0.08396
Market	2.8567	0.82888

Residual Scale Estimates:

```
oil.lm : 0.4867 on 127 degrees of freedom
oil.robust : 0.1446 on 127 degrees of freedom
```

### Visualizing LS vs. Robust Fits

You can easily plot a `compare.fits` object to obtain a visual comparison of the LS and robust fits:

```
> plot(oil.cmpr)
```

Make a plot selection (or 0 to exit):

- 1: Normal QQ-Plots of Residuals
  - 2: Estimated Densities of Residuals
  - 3: Residuals vs Fitted Values
  - 4: Response vs Fitted Values
- Selection:

For example, the normal QQ-plot and estimated densities for `oil.cmp` are shown in Figure 3. The densities of residuals are estimated using a kernel type density estimate. For a good model fit, the probability density estimates for the residuals will be centered at zero and nearly as narrow as possible. Figure 3 shows that the density of residuals from the LS estimate is shifted to the left of the origin, whereas that of the robust fit is well centered. Furthermore, the outlier bumps in the residual density estimates for the MM-estimator are pushed further from the mode of the density, and thus are a little more pronounced than those for the LS estimates (because there is one big outlier in the data).

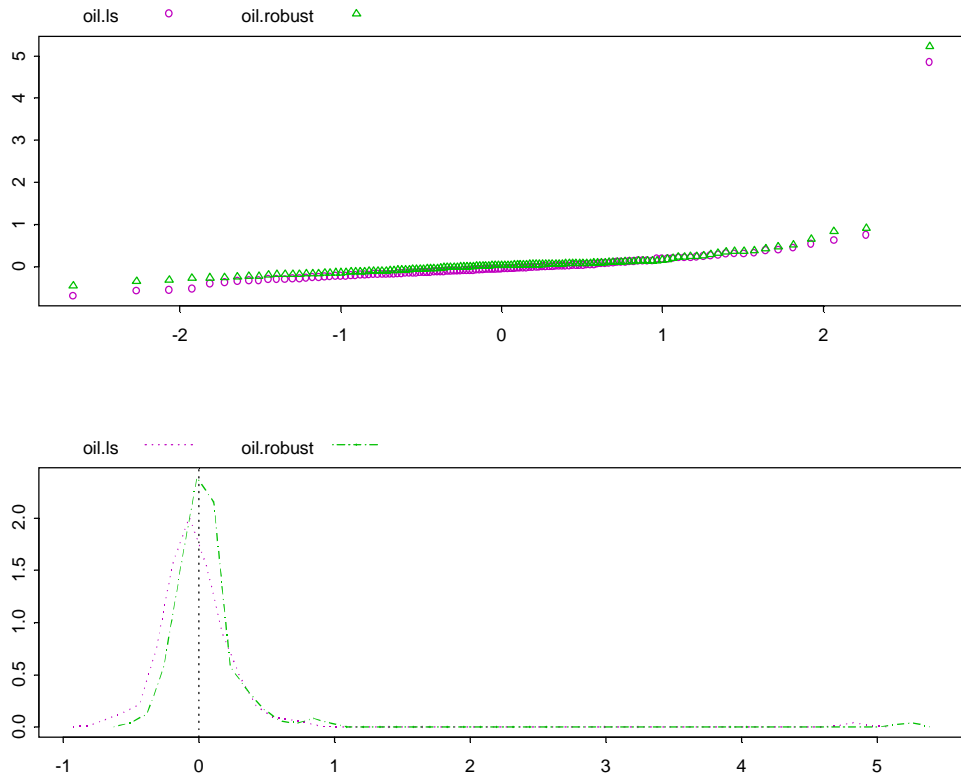


Figure 12.3: *Sample Plots of oil.l.cmpr*

## Statistical Inference for LS vs. Robust Fits

A more detailed comparison, particularly comparison of t-values and p-values, can be obtained using the generic summary function on a “compare.fits” object. For example:

```
> summary(oil.l.cmpr)
```

Call is:

```
oil.ls lm(formula = Oil ~ Market, data = oil.df)
```



```
oil.robust <- lmRobMM(formula = Oil ~ Market, data = oil.df)
```

Residual Statistics:

	Min	1Q	Median	3Q	Max
oil.lm	-0.6952	-0.17323	-0.05444	0.08407	4.842
oil.robust	-0.4566	-0.08875	0.03082	0.10314	5.218

Coefficients:

	Value	Std. Error	t value	Pr(> t )
(Intercept)	0.1474	0.07072	2.085	0.0390860
Market	2.8567	0.73175	3.904	0.0001528

oil.robust

	Value	Std. Error	t value	Pr(> t )
(Intercept)	-0.08396	0.02805	-2.993	0.003321
Market	0.82888	0.28342	2.925	0.004087

Residual Scale Estimates:

oil.lm : 0.4867 on 127 degrees of freedom  
oil.robust : 0.1446 on 127 degrees of freedom

Proportion of variation in response(s) explained by model(s):

oil.lm : 0.1071  
oil.robust : 0.05261

Correlations:

	Market
(Intercept)	0.7955736

oil.robust

	Market
(Intercept)	0.8168693

**Caveat:** When the final M-estimate is not used, i.e., p-values of test for bias indicates that the final M-estimate is highly biased relative to the initial S-estimates, the asymptotic approximations for the inference may not be very good and you should not trust them very much.

## ROBUST MODEL SELECTION

### Robust F and Wald Tests

Another important part of statistical inference is hypothesis testing. S-PLUS provides two robust tests for testing whether or not some of the regression coefficients are zero: the robust Wald test and the robust F test. For technical details on how these tests are computed, see the “*Theoretical Details of the Robust Regression Method*” below. Before proceeding, you will first create the data frame `si mu. dat`:

```
> si mu. dat <- gen. data(1: 3)
```

where the function `gen. data` is provided in the appendix. This function generates a data frame with five columns:  $y$ ,  $x_1$ ,  $x_2$ ,  $x_3$  and  $x_4$ . The variable  $y$  is generated according to the following equation:

$$y = b_1x_1 + b_2x_2 + b_3x_3 + u$$

where  $b_1, b_2, b_3$  is given by 1: 3 in the above S-PLUS command, and  $u$  is sampled from a  $N(0,3)$  family with 10% contamination. The term  $x_4$  is independent of  $y$ ,  $x_1$ ,  $x_2$  and  $x_3$ . First, you fit a model with  $x_1$ ,  $x_2$ ,  $x_3$  and  $x_4$  as the predictor variables:

```
> si mu. mm4 <- lmRobMM(y~x1+x2+x3+x4-1, data=si mu. dat)
```

To test the hypothesis that the coefficient of  $x_4$  is actually zero, you can fit another model with only  $x_1$ ,  $x_2$  and  $x_3$  as the predictor variables, then use `anova` to test the significance of the coefficient of  $x_4$ :

```
> si mu. mm3 <- update(si mu. mm4, . ~. -x4)
```

```
> anova(si mu. mm4, si mu. mm3)
```

Response: y

	Terms	Df	Wal d	P(>Wal d)
1	x1 + x2 + x3 + x4 - 1			
2	x1 + x2 + x3 - 1	1	0.04436687	0.8331725

The  $p$ -value in this case is greater than 0.8, which leads you to accept the null hypothesis that the fourth coefficient value is zero.

The default test used by `anova` is the Wald test based on robust estimates of the coefficients and covariance matrix (a robust Wald test). To use the robust F test instead, specify the optional argument `test` to `anova`:

```
> anova(simu.mm4, simu.mm3, test="RF")
```

```
Response: y
```

	Terms	Df	RobustF	P(>RobustF)
1	x1 + x2 + x3 + x4 - 1			
2	x1 + x2 + x3 - 1	1	0.03374514	0.8507404

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

## Robust FPE Criterion

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, S-PLUS provides Robust Final Prediction Errors (RFPE) as a criterion, which is a robust analogue to the classical Final Prediction Errors (FPE) criterion. RFPE is defined as:

$$RFPE = \sum_{i=1}^n E \rho \left( \frac{y_i^* - x_i^T \beta^{(1)}}{\sigma} \right),$$

where  $\beta^{(1)}$  is the final M-estimate of  $\beta$ ,  $y_i^*$ 's are the values you are trying to predict using  $\beta^{(1)}$ , and the expectation is taken with respect to both  $\beta^{(1)}$  and  $y_i^*$ 's. When considering a variety of model choices with respect to different choices of predictor variables, you choose the model with the smallest value of RFPE.

Note that when  $\rho(u) = u^2$ , RFPE reduces to the classical FPE. RFPE can also be shown to be asymptotically equivalent to the robust version of AIC proposed by Ronchetti (1985). The section *Theoretical Details of the Robust Regression Method* provides a sketch of technical details supporting the use of RFPE.

The RFPE criterion is used as the robust method, invoked by use of the generic functions, of `drop1` and `add1`. For example, use of `drop1` on the robustly fitted model `simu.mm4` in the previous section gives:

```
> drop1(simu.mm4)
```

```
Significant test at level 10 %
for x3
```

## Single term deletions

Model :

 $y \sim x1 + x2 + x3 + x4 - 1$ 

	Df	RFPE
<none>		24.24174
x1 1		24.46596
x2 1		52.19800
x3 1		64.32633
x4 1		23.95825

The output indicates that dropping  $x4$  gives a better model.

You can also use `add1` to explore the relevance of other variables. For example, if you fit `si mu. mm3` first, you can use the following command to investigate if  $x4$  helps predict  $y$ .

```
> add1(si mu. mm3, "x4")  
Single term additions
```

Model :

 $y \sim x1 + x2 + x3 - 1$ 

	Df	RFPE
<none>		24.10184
x4 1		24.38769

Since addition of  $x4$  causes RFPE to increase, addition of  $x4$  results in a poor model.

**Caveat:** If the test for bias of final M-estimates is significant for any of the models considered by `drop1` and `add1`, you should not trust the corresponding RFPE very much.

# CONTROLLING OPTIONS FOR ROBUST REGRESSION

In this subsection, you will learn how to change the default settings of some control parameters for the MM-estimator so as to obtain particular estimates that fit your purpose. Most of the default settings can be changed through the functions `lm.robust.control` and `lm.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 `lm.robust.control` and `lm.genetic.control`.

## Efficiency at Gaussian Model

If the final M-estimates are accepted, they have a default asymptotic efficiency of 85% compared with the LS estimates, when the errors are normally distributed.

Sometimes an asymptotic efficiency of 85% may not be what you exactly want. To change the efficiency of the final M-estimates, the `lmRobMM` optional argument `robust.control` should be generated from `lmRobMM.robust.control` with desired efficiency:

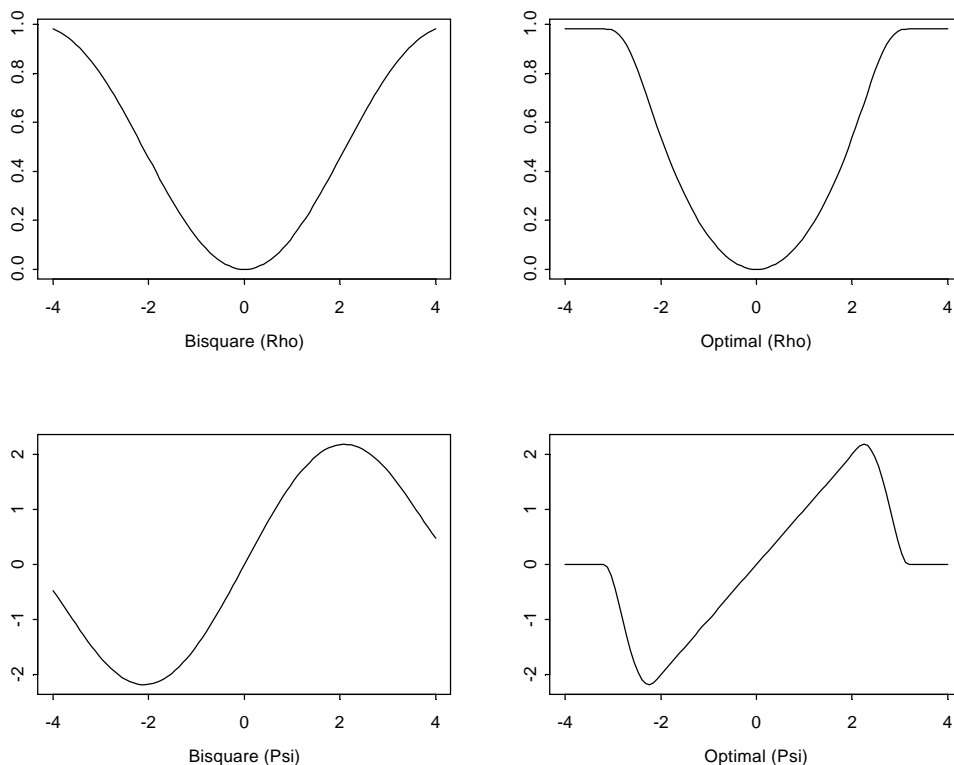
```
> oil.tmp <- lmRobMM(Oil ~ Market, data=oil.df,
+   robust.control=lmRobMM.robust.control(efficiency=0.95))

> coef(oil.tmp)
(Intercept)      Market 
 -0.07398806  0.8491126
```

## Alternative Loss Function

As mentioned in the introduction, the final M-estimates are based on the initial S-estimates of regression coefficients and scale parameter. For both the initial S-estimate and the final M-estimate, S-PLUS uses a loss function for the estimation. Two different loss functions are available in S-PLUS: Tukey's bisquare function and the optimal loss function recently discovered by Yohai and Zamar (1998). Figure 4 shows the Tukey bisquare function on the left

and the optimal loss function on the right.



**Figure 12.4:** *Available Loss Functions*

The exact forms of these functions can be found in the *Theoretical Details* section.

Since the optimal loss function above has better combined Gaussian efficiency and non-Gaussian bias control properties, it is used as the default for robust regression. However, you can choose to use the Tukey bisquare function or a combination of those two functions by controlling the `weight` argument to `lmRobMM.robust.control` as follows:

```
> control <- lmRobMM.robust.control(weight=c("Bisquare",
+                                           "Optimal"))
```

```

> oil.tmp <- lmRobMM(Oil ~ Market, data=oil.df,
+   robust.control = control)

> coef(oil.tmp)
(Intercept)      Market
-0.08371818  0.8291069

```

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.

## Confidence Level of Bias Test

In the `oil.robust` example shown above, the final M-estimates are accepted over the initial S-estimates because the  $p$ -value of the test for bias is 0.33. The default level of this test is set at 10%, so whenever the  $p$ -value of the test is greater than 10%, the final M-estimates are returned; otherwise, the initial S-estimates are returned.

To change the level of the test for bias of the final M-estimates to a different value, you should specify the argument `level` for the `lmRobMM.robust.control` 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. For example, you can force the procedure to return the initial S-estimates by using the following commands:

```

> control.s <- lmRobMM.robust.control(level=1)

> oil.s <- lmRobMM(Oil ~ Market, data=oil.df,
+   robust.control = control.s)
Significant test at level 100 %
> oil.s
Initial S-estimates.

```

```

Call:
lmRobMM(formula = Oil ~ Market, data = oil.df,
  robust.control = control.s)

```

```

Coefficients:
(Intercept)      market
-0.06244374  0.8273216

```

```

Degrees of freedom: 129 total; 127 residual
Residual scale estimate: 0.1446283

```

**Warning:** The bias is high; inference based on final estimates is not recommended; use initial estimates as exploratory tools.

**Caveat:** The above warning is only relevant when you use levels in the range of 1% to 10%, and the choice of level in this range is a rather subjective choice of the user.

Similarly, using `level = 0` forces `lmRobmm` to return the final M-estimates:

```
> control.mm <- lmRobMM.robust.control(level = 0)

> oil.mm <- lmRobMM(Oil ~ Market, data = oil.df,
+                   robust.control = control.mm)
```

Sometimes you may want to change the level of the test after fitting a robust regression model. For this purpose, you can use the generic function `update`, which has a method for "`lmRobMM`" objects. For example, to change the level of test for bias for `oil.s`, use the following command:

```
> oil.tmp <- update(oil.s, level = 0.2)

> oil.tmp
Final M-estimates.

Call:
lmRobMM(formula = Oil ~ Market, data = oil.df,
         robust.control = control.s)

Coefficients:
(Intercept)      Market 
-0.08395777  0.8288791 

Degrees of freedom: 129 total; 127 residual
Residual scale estimate: 0.1478398
```

Now the final M-estimates are returned. Also, if both the `formula` and the `level` arguments are missing for `update`, the function alternates between the initial S-estimates and final M-estimates.

**Note:** If you only want to compute the S-estimates and do not care about the final M-estimates, you can do so by specifying the `estim` argument to `lmRobMM.robust.control` as follows:

```
> control.s <- lmRobMM.robust.control(estim = "S")

> oil.s <- lmRobMM(Oil ~ Market, data = oil.df,
+                 robust.control = control.s)
```



```

> oil.s
Initial S-estimates.

Call:
lmRobMM(formula = Oil ~ Market, data = oil.df,
         robust.control = control.s)

Coefficients:
(Intercept)      Market
   -0.06244374    0.8273216

Degrees of freedom: 129 total; 127 residual
Residual scale estimate: 0.1446283

```

Similarly, you can get the final M-estimates if you use `estim="MM"`.

## Resampling Algorithms

When computing the initial S-estimates, a resampling scheme is used. S-PLUS provides three resampling algorithms for the initial S-estimates: random resampling, exhaustive resampling and genetic algorithm. These algorithms can be selected by using the `sampling` argument to the function `lmRobMM.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  $\lceil 4.6 \cdot 2^p \rceil$ , where  $p$  is the number of explanatory variables, and  $\lceil \cdot \rceil$  denotes the operation of rounding a number to its closest integer. Note that this number will work fine if you have less than 13 predictor variables. However, if you have more than 13 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 <- lmRobMM(Oil ~ Market, data=oil.df, nrep=10)
```

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

## Genetic Algorithm Parameters

If you choose to use the genetic algorithm, the parameters for genetic algorithm can be changed through the `lmRobMM` 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 `lmRobMM.genetic.control`. To look at the arguments of the function `lmRobMM.genetic.control`, use the following command:

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

For an explanation of the various arguments above, you should read the help file for the function `ltsreg.default`.

## THEORETICAL DETAILS

### Initial Estimate Details

The key to obtaining a good local minimum of the M-estimation objective function when using a bounded, non-convex loss function is to compute a highly robust initial estimate  $\beta^0$ . S-PLUS does this by using the S-estimate method introduced by Rousseeuw and Yohai (1984), as 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  $y_1, y_2, \dots, y_n$ , assumed to be robustly centered (i.e., by subtracting a robust location estimate). The M-estimate  $\hat{s}$  is obtained by solving the equation

$$\frac{1}{n} \sum_{i=1}^n \rho\left(\frac{y_i}{\hat{s}}\right) = .5 \quad (12.1)$$

where  $\rho$  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 (3.7):

$$\frac{1}{n-p} \sum_{i=1}^n \rho\left(\frac{y_i - x_i^T \beta}{\hat{s}(\beta)}\right) = .5 \quad (12.2)$$

For each value of  $\beta$  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  $\hat{s}(\beta)$  :

$$\hat{\beta}^0 = \arg \min_{\beta} \hat{s}(\beta) \quad (12.3)$$

This presents another non-linear 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). S-PLUS 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 l mRobMM.

## Optimal and Bisquare Rho and Psi-Functions

A robust M-estimate of regression coefficient  $\beta$  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 denoted by  $\psi(\cdot; c)$ . 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:

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

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

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

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

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

where  $g_1 = -1.944$ ,  $g_2 = 1.728$ ,  $g_3 = -0.312$ ,  $g_4 = 0.016$ , and

$$h_1 = \frac{g_1}{2}, h_2 = \frac{g_2}{4}, h_3 = \frac{g_3}{6}, h_4 = \frac{g_4}{8}.$$

## The Efficient Bias Robust Estimate

Yohai and Zamar (1998) showed that the  $\rho$  and  $\psi$  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 is 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 S-PLUS will automatically use 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  $\mu$ . 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  $\mu$  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  $\mu^*$  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  $\sigma^2$ -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

See chapter 7 of Hampel, Ronchetti, Rousseeuw and Stahel (1986), where this test is referred to as the “tau” test.

## 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. However, the results therein are subject to certain restrictions, such as they only apply to M-estimates with zero breakdown point and the density of the errors has to be in a certain form. Yohai (1997) proposed the following RFPE criterion which is not subject to the restrictions that apply to Ronchetti's robust version of AIC.

$$RFPE = nE\rho\left(\frac{\varepsilon}{\sigma}\right) + p\frac{A}{2B} \quad (12.4)$$

where

$$A = E\Psi^2\left(\frac{\varepsilon}{\sigma}\right) \quad B = E\Psi'\left(\frac{\varepsilon}{\sigma}\right).$$

Since the first term in equation (12.4) can be approximated by

$$nE\rho\left(\frac{\varepsilon}{\sigma}\right) \approx \sum_{i=1}^n \rho\left(\frac{r_i}{\sigma}\right) + p \frac{A}{2B}$$

where  $r_i = y_i - x_i^T \hat{\beta}^1$ , the expression (12.4) can be estimated by

$$RFPE \approx \left[ \sum_{i=1}^n \rho\left(\frac{y_i - x_i^T \hat{\beta}^1}{\hat{s}^{(0)}}\right) \right] + p \frac{\hat{A}}{\hat{B}} \quad (12.5)$$

with

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

The approximation on the right hand side of (12.5) is used as our RFPE.

## Appendix

The function `gen.data` used in the section *Robust Model Selection* is as follows:

```
> gen.data <- function(coeff, n = 100, eps = 0.1, sig = 3,
+   snr = 1/20, seed = 837)
+ {
+   # coeff : 3 x 1 vector of coefficients
+   # eps   : the contamination ratio, between 0 and 0.5
+   # sig   : standard deviation of most observations
+   # snr   : signal-to-noise ratio, well, not really
+   # Note  : the regressors are generated as: rnorm(n,1),
+   #         rnorm(n,1)^3, exp(rnorm(n,1)). It also generates
+   #         an unused vector x4.
+   set.seed(seed)
+   x <- cbind(rnorm(n, 1), rnorm(n, 1)^3,
+             exp(rnorm(n, 1)))
+   ru <- runif(n)
+   n1 <- sum(ru < eps)
+   u <- numeric(n)
+   u[ru < eps] <- rnorm(n1, sd = sig/snr)
+   u[ru > eps] <- rnorm(n - n1, sd = sig)
+   data.frame(y = x %*% matrix(coeff, ncol = 1) + u,
```



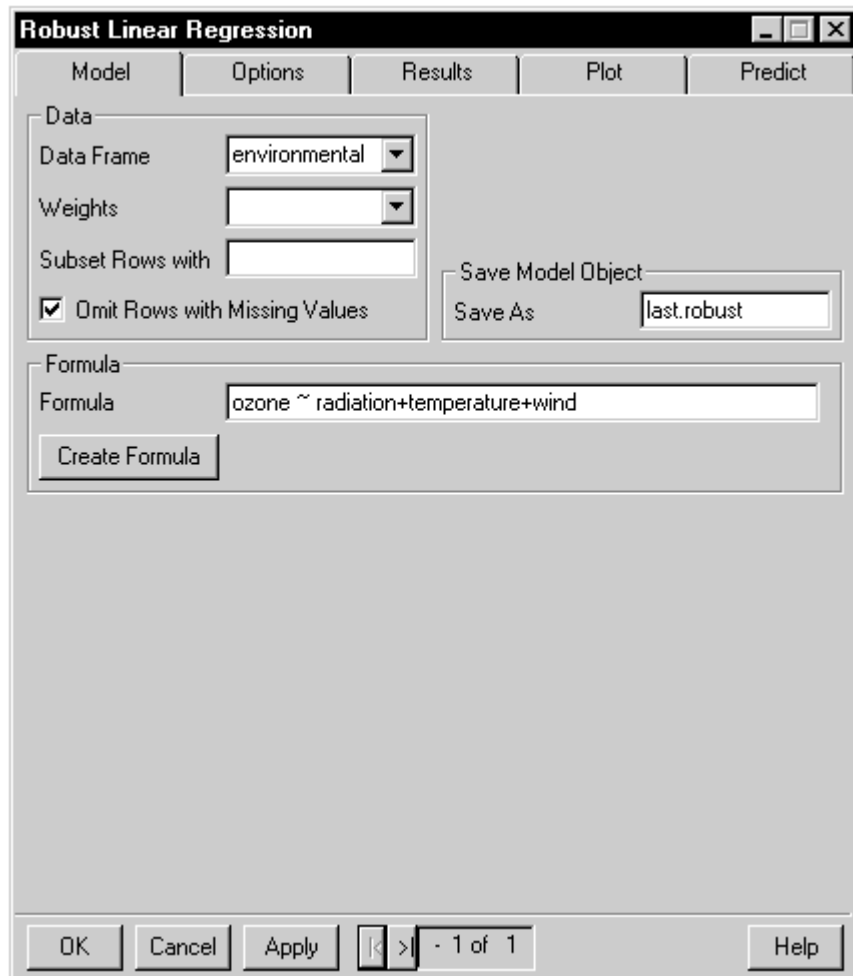
```
+      x1 = x[, 1], x2 = x[, 2], x3 = x[, 3],  
+      x4 = rnorm(n, 1))  
+ }
```

## ROBUST MM REGRESSION

This dialog fits linear regression models using a robust method based on the collective work of Rousseeuw and Yohai (1984), Yohai, Stahel, and Zamar (1991), Marazzi (1993), and Yohai and Zamar (1998). It calls the `lmRobMM` function and its `print`, `summary`, `plot` and `predict` methods.

**To perform linear regression:**

Choose **Statistics:Regression:RobustMM** from the main menu. The dialog shown below appears.



The image shows a software dialog box titled "Robust Linear Regression". It has five tabs: "Model", "Options", "Results", "Plot", and "Predict". The "Options" tab is currently selected. Under the "Data" section, there are three dropdown menus: "Data Frame" (set to "environmental"), "Weights" (empty), and "Subset Rows with" (empty). There is a checked checkbox labeled "Omit Rows with Missing Values". To the right of these is a "Save Model Object" section with a "Save As" label and a text box containing "last.robust". Under the "Formula" section, there is a text box containing the formula "ozone ~ radiation+temperature+wind" and a "Create Formula" button. At the bottom of the dialog are buttons for "OK", "Cancel", "Apply", a page indicator showing "< > 1 of 1", and a "Help" button.

{bmc robmm1.bmp}

## Model Page

**Data** **Data Frame**  
Select a data frame.

### Tip...

*You can type into the Data Frame edit box any expression which evaluates to a data frame.*

**Weights**

Enter the column that specifies weights to be applied to all observations used in the linear regression. To weigh all rows equally, leave this blank.

**Subset Rows with**

Enter an S-PLUS expression which identifies the rows to use in the analysis. To use all the rows in the data frame, leave this field blank. The expression must evaluate to a vector of logical values (TRUE values are used, FALSE values are dropped), or a vector of indices identifying the numbers of the rows to use.

Examples:

`Species == 'bear'`                      only bears are used.

`1:20`                                      only the first 20 rows of the data are used.

`Age >= 13 & Age < 20`                  only teenagers are used.

For more information on constructing logical expressions see the S-PLUS *Programmer's Guide*.

**Omit Rows with Missing Values**

Check this box to omit from the analysis any rows in the data frame that contain missing values for any of the variables in the model.

If this box is not checked, S-PLUS will report an error and halt the routine if any row is found to have a missing value in any of the terms in the model.

**Formula Formula**

Enter a formula specifying the desired model. The formula specifies which regression model is to be fit. In its simplest form a formula consists of the response variable, a tilde (~), and a list of predictor variables separated by “+”s. An intercept is automatically included by default. For example: `Fuel ~ Weight + Disp.` fits a regression model with Fuel as the response and Weight and Disp. as predictors.

For more information on formulas see the chapter on Building Formulas.

**Create Formula**

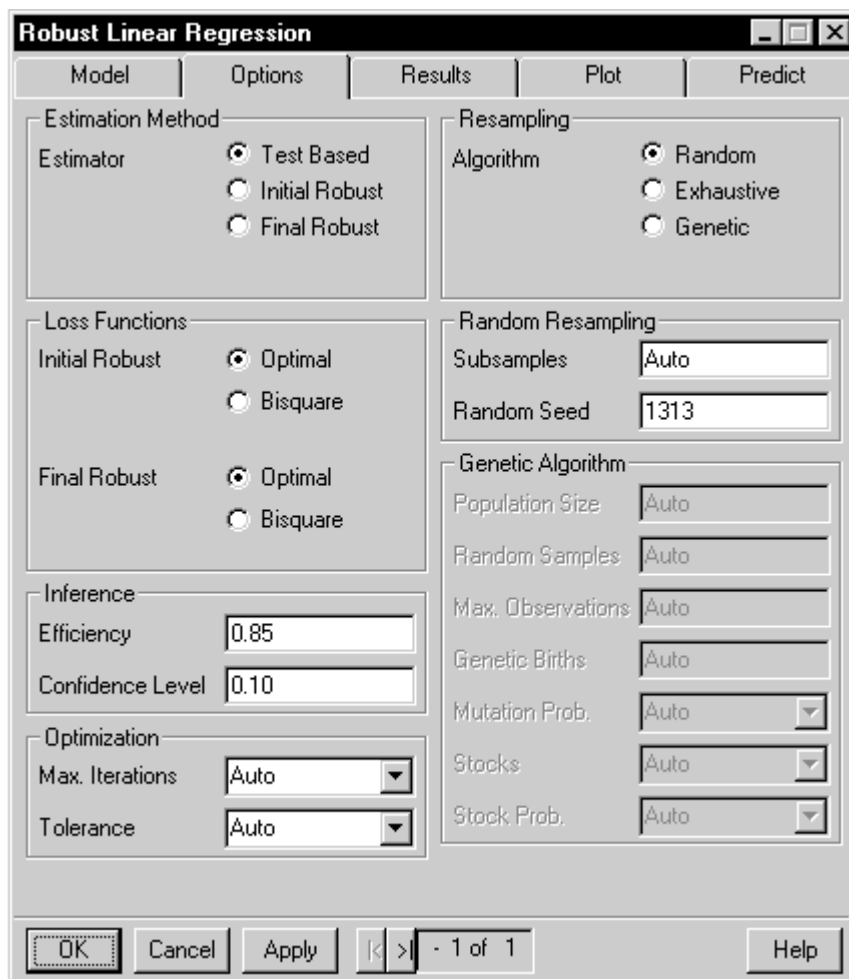
Click this to open a formula builder dialog used to construct a formula specifying the desired model. See the chapter Building Formulas for more information.

**Save Model Object Save As**

Enter the name for the object in which to save the results of the analysis. If an object with this name already exists, its contents will be overwritten.

This must be a valid S-PLUS object name—any combination of alphanumeric characters that starts with an alpha character is allowed. The only non-alphanumeric character allowed is the period “.”. Names are case-sensitive, so X and x are different names. The default is “last.robust”.

The saved object will have class “l mRobMM”. See the on-line help for l mRobMM. obj ect for more information about the saved object.



{bmc robmm2.bmp}

**Options Page** The Options page of the Robust Linear Regression dialog contains the controls specific to the robust MM estimate regression method.

#### Estimation Method **Test Based**

Select this option to have S-PLUS calculate both initial S-estimates and final M-estimates, and report results for one of these estimates, based on a test for bias.

**Initial Robust**

Select this option to have S-PLUS calculate only initial S-estimates.

**Final Robust**

Select this option to have S-PLUS calculate only final M-estimates.

**Loss Functions Initial Robust**

Select the desired loss function, "Optimal" or "Bisquare", for the initial S-estimates.

**Final Robust**

Select the desired loss function, "Optimal" or "Bisquare", for the final M-estimates.

**Inference Efficiency**

Enter the asymptotic efficiency of the final M-estimates.

**Confidence Level**

Enter the desired level of significance of the test for bias of the final M-estimates.

**Optimization Max. Iterations**

Enter the name of a list with components "mxf", "mxr", and "mxs", representing the maximum number of iterations, respectively, for final coefficient estimates, the refinement step, and the final scale estimate. The default value, Auto, sets all three to 50.

**Tolerance**

Enter the name of a list with components "tlo", "tua", and "tl", representing, respectively, the relative tolerance in the iterative algorithms, the tolerance used for the determination of pseudo-rank, and the tolerance for scale denominators. The default value, Auto, sets the tolerances as follows:

```
tlo = 0.0001, tua = 1.5e-006, tl = 1e-006
```

**Resampling Random**

Select this option to use the random resampling algorithm.

**Exhaustive**

Select this option to use the exhaustive resampling algorithm only if the sample size is less than 300 and the number of predictor variables is less than 10.

**Genetic**

Select this option to use the genetic resampling algorithm.

**Random  
Resampling Subsamples**

Enter the number of random subsamples to be drawn. The default value, Auto, draws  $4.6 \times 2^{\text{ncol}(x)}$  samples.

**Random Seed**

Enter the seed parameter used in the random sampling algorithm.

**Genetic Algorithm Population Size**

Enter the population size of the genetic stock. The default is 10 times the number of parameters being fit.

**Random Samples**

Enter the number of random samples taken after the stock is filled. The default is 50 times the number of parameters being fit.

**Max Observations**

Enter the maximum number of observations (including duplicates) in a member of the stock. The default is  $p$  if  $(n-p)/2$  is less than  $p$ , where  $n$  is the number of observations; otherwise it is the minimum of  $\text{trunc}((n-p)/2)$  and  $5 \cdot p$ .

**Genetic Births**

Enter the number of genetic births. The default is  $(50 \cdot p) + (15 \cdot p^2)$ .

**Mutation Prob.**

Enter a length 4 vector of mutation probabilities for offspring.

**Stocks**

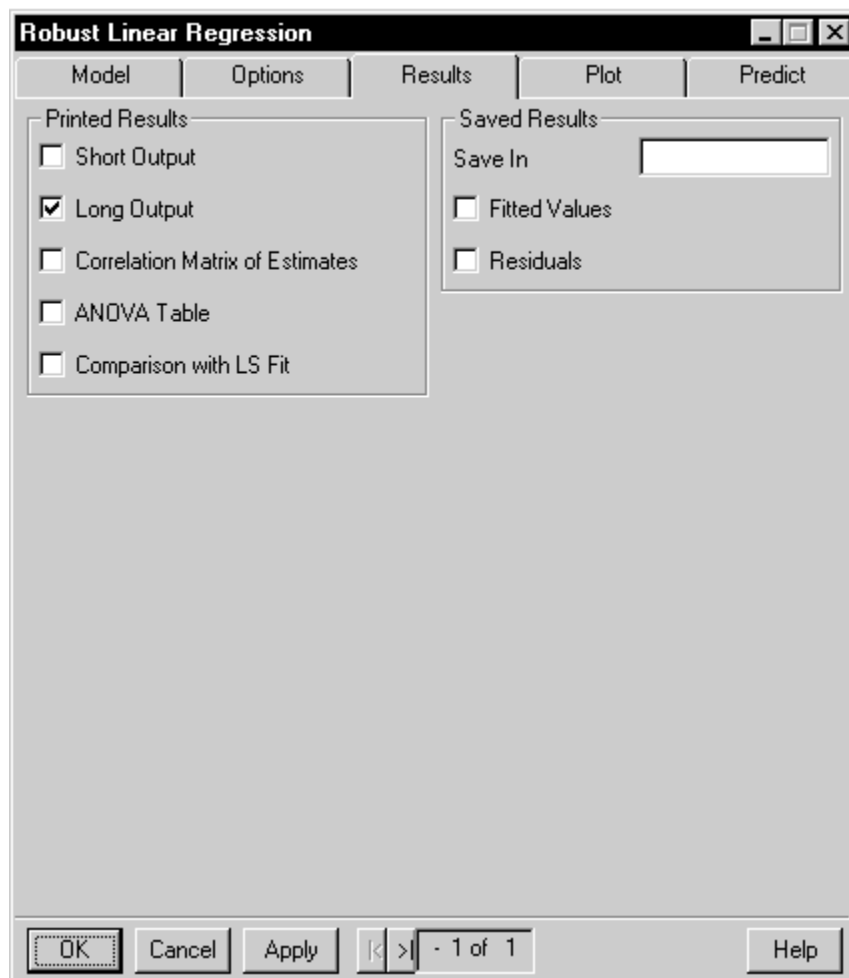
Enter a list of vector of observation numbers to be included in the stock. This is typically the `stock` component of the output of a previous run.

**Stock Prob.**

Enter a vector of cumulative probabilities that a member of the stock will be chosen as a parent. The  $i$ th element corresponds to the individual with the  $i$ th lowest objective. The default is

```
cumsum((2*(popsize: 1))/popsize/(popsize+1))
```

## Results Page



### Printed Results Short Output

Check here to display a short summary of the model fit. This includes the model formula, the robust estimates of regression coefficients and residual scale, and the degrees of freedom.

### Long Output

Check here to display a detailed summary of the model fit. This includes the model formula; a five number summary of the residuals; the robust estimates of coefficients; robust standard errors, robust t-statistics and robust p-values; the robust residual scale estimate and the degrees of freedom; the robust multiple R-Squared value; a test for bias; and the seed parameter.



**Correlation Matrix of Estimates**

Check here to display the robust correlation matrix of the regression coefficients. This is only available if the Long Output is selected.

**ANOVA Table**

Check here to display an analysis of variance table. The sums-of-squares in the table are for the terms added sequentially (Type I sums-of-squares).

**Comparison with LS Fit**

Check here to display the output of the robust MM-estimate fit together with the results for a standard least squares linear model fit of the same formula.

**Saved Results Save In**

Enter the name of an S-PLUS data frame in which fitted values and residuals of the analysis are to be saved. If an object with the name you enter does not already exist (in database 1), then it will be created. If you enter the name of a data frame that already exists (in database 1) and this data frame has the same number of rows as the number of observations used in the model fit, then the saved values are appended to this data frame. This allows you to keep fitted values from a model with the original data or to keep the residuals from a number of different models for the same data in one data frame. If you give the name of an existing S-PLUS object that is not a data frame or is not the appropriate size, then a warning is issued and a modified name is used.

**Tip...**

*You may want to specify the same data frame as on the Model page. This allows easy plotting of the fitted values or residuals with the original data.*

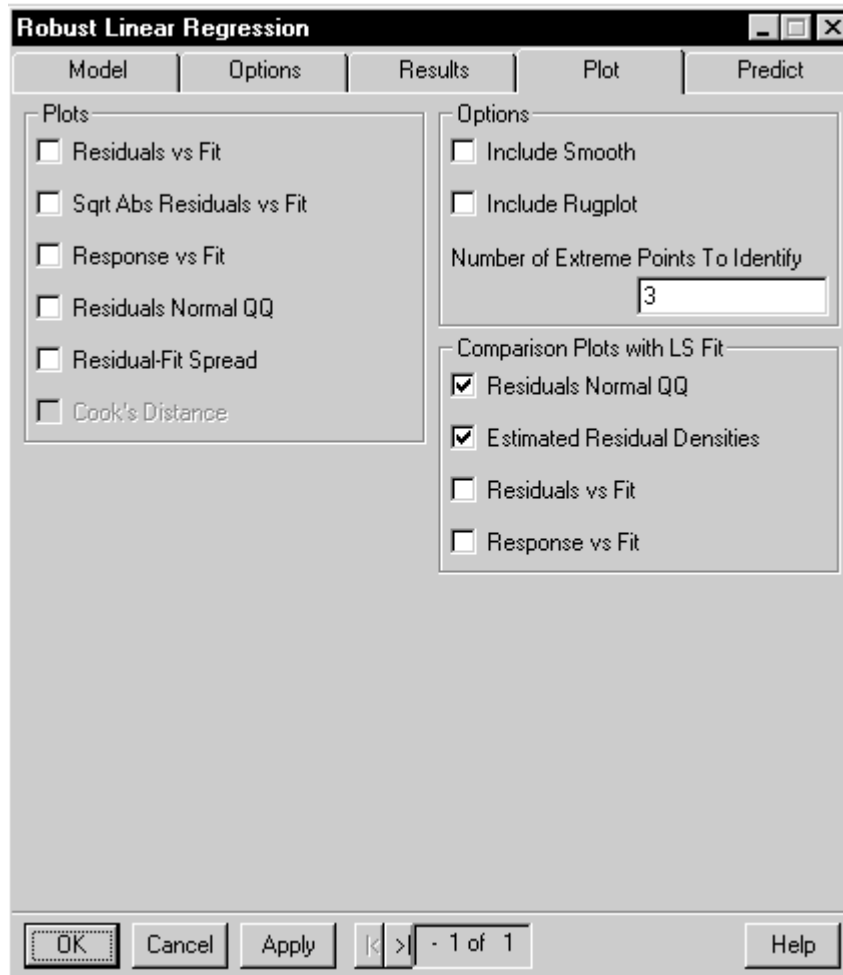
**Fitted Values**

Check this to save the fitted values from the model in the object specified in Save In.

**Residuals**

Check this to save the residuals from the model in the object specified in Save

In. These are the ordinary residuals; the response minus the fitted value.



{bmc linreg3.bmp}

## Plot Page

### Plots Residuals vs Fit

Check this to display a plot of the residuals versus the fitted values.

### Sqrt Abs Residuals vs Fit

Check this to display a plot of the square root of the absolute values of the residuals versus the fitted values. This plot is useful for checking for the constant variance assumption of the model.

**Response vs Fit**

Check this to display a plot of the response variable versus the fitted values. The line  $y = x$  is also drawn on the graph.

**Residuals Normal QQ**

Check this to display a Normal quantile-quantile plot of the residuals.

**Residual-Fit Spread**

Check this to display a residual-fit spread plot. This is a visual analog of the multiple R-squared statistic. It compares the spread of the fitted values to the spread of the residuals.

**Cook's Distance**

This plot is not available for the robust MM model.

**Options Include Smooth**

Check this to display a smooth curve, computed with `loess.smooth`, on the Residuals vs Fit, Sqrt Abs Residuals vs Fit, and Response vs Fit plots. See the on-line help for `loess.smooth` for details.

**Include Rugplot**

Check this to display a rugplot on the Residuals vs Fit, Sqrt Abs Residuals vs Fit, and Response vs Fit plots. A rugplot is a sequence of vertical bars along the x-axis that mark the “observed” x values.

**Number of Extreme Points to Identify**

Enter the number of extreme points that will be identified on the Residuals vs Fit, Sqrt Abs Residuals vs Fit, and Residuals Normal QQ. The row names from the data frame specified on the model page will be used to identify the points.

**Comparison Plots Residuals Normal QQ****with LS Fit**

Check this to include a graph showing the qqnorm plot of the residuals of the robust fit together with the qqnorm plot of the residuals of the standard least squares fit.

**Estimated Residual Densities**

Check this to include a graph showing the density estimate for the residuals of the robust fit together with the density estimate for the residuals of the standard least squares fit.

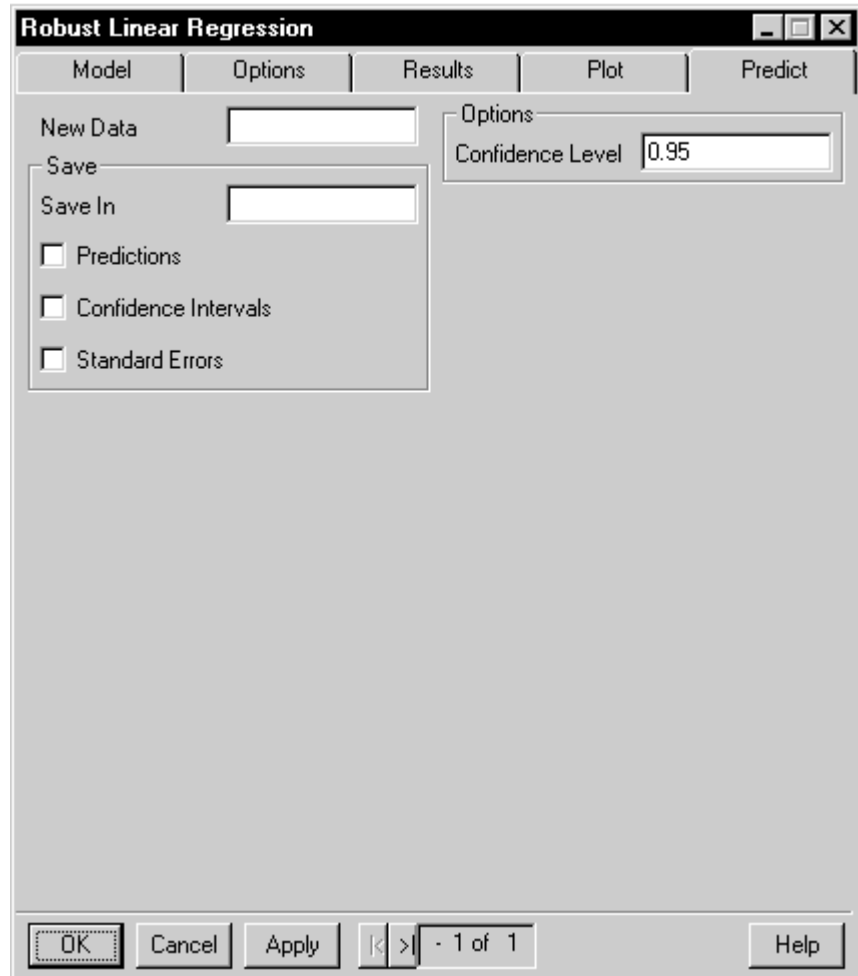
**Residuals vs Fit**

Check this to include a graph showing the residuals vs fit plots for both the robust model and the standard least squares fit.

**Response vs Fit**

Check this to include a graph showing the response vs fit plots for both the

robust model and the standard least squares fit.



{bmc linreg4.bmp}

### Predict Page **New Data**

Enter the name of a data frame to use for computing predictions. It must contain the same names as the terms in the right side of the formula for the model. If omitted, the original data are used for computing predictions.

### Save **Save In**

Enter the name of an S-PLUS data frame in which predictions, confidence intervals and standard errors are to be saved. If an object with the name you enter does not already exist (in database 1), then it will be created. If you

enter the name of a data frame that already exists (in database 1) and this data frame has the same number of rows as the number of observations used in the model fit, then the saved values are appended to this data frame. This allows you to keep predicted values from a model with the original data or to keep the residuals from a number of different models for the same data in one data frame. If you give the name of an existing S-PLUS object that is not a data frame or is not the appropriate size, then a warning is issued and a modified name is used.

#### **Predictions**

Check this to save the predictions in the data frame specified in Save In.

#### **Confidence Intervals**

Check this to store lower and upper confidence limits in the object specified in Save In. The column names will be " $N\%$  L.C.L." and " $N\%$  U.C.L." where  $N$  is 100 times the value specified in Confidence Level. These confidence limits for the mean response are computed as the prediction plus or minus  $t$ -value times standard error.

#### **Standard Errors**

Check this to store the pointwise standard errors for the predictions in the object specified in Save In.

### **Options Confidence Level**

Enter the confidence level to use when computing confidence intervals. This value should be less than 1 and greater than 0.

### **S-PLUS language functions related to Linear Models:**

```
lmRobMM, plot.lmRobMM, predict.lm, print.lmRobMM,
summary.lmRobMM, lmRobMM.robust.control,
lmRobMM.genetic.control
```

### **Other related S-Plus language functions:**

```
aov, gam, glm, lm, loess, nls
```

## BIBLIOGRAPHY

- Hampel, F., Ronchetti, E. M., Rousseeuw, P. J. and Stahel, W. A. (1986): *Robust Statistics: the Approach Based on Influence Functions*, John Wiley & Sons.
- Huber, P.J. (1981). *Robust Statistics*. John Wiley & Sons.
- Marazzi, A. (1993): *Algorithms, Routines, and S Functions for Robust Statistics*, Wadsworth & Brooks/Cole, Pacific Grove, CA.
- Martin, R. D. and Zamar, R. H. (1989). Asymptotically Min-Max Robust M-estimates of Scale for Positive Random Variables. *J. Amer. Statist. Assoc.*, 84, 494-501.
- Martin, R. D. and Zamar, R. H. (1993). Bias Robust Estimates of Scale. *Annals of Statistics*.
- Ronchetti, E. (1985): Robust Model Selection in Regression, *S-PLUS Statistics & Probability Letters*, 3, 21--23.
- 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, and R. D. Martin (eds.), *Lecture Notes in Statistics*, 26, 256-272, Springer-Verlag.
- 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 Stahel, W. A. and Weisberg, S. W., Eds., *Directions in Robust Statistics and Diagnostics, Part II*, Springer-Verlag, New York.
- Yohai, V. J. and Zamar (1998). "Optimal locally robust M-estimates of regression", *Jour. of Statist. Inf. and Planning*.

# PARAMETRIC REGRESSION FOR CENSORED DATA

# 13

---

<b>Introduction</b>	<b>196</b>
<b>The Generalized Kaplan-Meier Estimate</b>	<b>198</b>
Specifying Interval Censored Data	198
Computing Kaplan-Meier Estimates	200
<b>ensorReg</b>	<b>203</b>
An Example Model	203
Specifying the Parametric Family	204
Accounting for Covariates	206
Truncation Distributions	208
Threshold Parameter	210
Offsets	211
Fixing parameters	212
<b>Fitting Models: ANOVA</b>	<b>214</b>
<b>Fitting Models: The plot method for CensorReg</b>	<b>216</b>
<b>Computing Probabilities and Quantiles</b>	<b>221</b>
<b>Parametric Survival</b>	<b>223</b>
Model Page	224
Options Page	226
Results Page	227
Plots Page	229
Predict Page	231

## INTRODUCTION

Parametric regression models for censored data are used in a variety of contexts ranging from manufacturing to studies of environmental contaminants. Because of their frequent use for modeling failure time or *survival* data they are often referred to as parametric survival models. In this context they are used throughout engineering to discover reasons why engineered products fail. They are called *accelerated failure time* models or *accelerated testing* models when the product is tested under more extreme conditions than normal to *accelerate* its failure time. Most product engineering can't wait long enough to observe ample failures for fitting models under normal operating conditions. The results obtained under extreme conditions are related to the results that *would* be obtained when the product is subject to normal wear. Thus, for example, capacitors may be operated under higher temperatures and voltages than normal to increase their likelihood of failure. The resulting fitted model is used to extrapolate failure rates back to normal operating conditions. Similar use is made of these failure time distributions in the context of *survival analysis* where living organisms rather than engineered products are the primary interest.

In the context of environmental studies, the measures of interest may be chemical contaminant levels rather than failure times but these data are frequently censored or obtained from truncated distributions. Censored and/or truncated data regression methodology applies equally well in these cases but, of course, the values of interest have nothing to do with survival.

Model selection is a major concern when using censored regression models. As in other model fitting activities, the distributional assumptions that are made must be appropriate for the data collected, and the model must also reasonably account for variation in the independent variables. Consequently, visual comparisons of the predicted (from the model) distribution of the response with nonparametric estimates of the distribution is an important activity when fitting models. To obtain the most appropriate model, usually a number of models with different failure distributions and/or dependence relationships with the independent variables will be fitted and compared. Visual comparison and statistical tests are then used to determine the most appropriate model.

Given that a model has been obtained, the results may be extrapolated to new values for the independent variables, and inference procedures may be used to obtain interval estimates for failure probabilities or quantiles of the response. In doing this, the usual precautions apply: one should not try to extrapolate model information to far beyond the values collected in the data. Moreover, because the interval estimate procedures are asymptotic, the confidence levels should be treated as approximate, especially in small



samples.

In this chapter we discuss a set of functions for the analysis of censored and/or truncated data or, more specifically, for the analysis of accelerated failure time and survival data. These functions are based upon estimation code originally developed by Meeker and Duke (1981) and refined subsequently by W. Q. Meeker (personal communication). This estimation code has been modified slightly for inclusion in the S-PLUS product. The S-PLUS code which calls the underlying estimation routines borrows from work done by both W. Q. Meeker and Terry Therneau. Taken as a whole, these functions allow you to easily specify and fit censored data models and to graph and compare the fitted models with appropriate non-parametric estimates of these models. You can also easily make inferences regarding the model parameters, predicted failure probabilities, and quantiles. We begin by briefly discussing the non-parametric estimates and how they may be computed. This brief introduction is followed by the meat of the software - a complete discussion of the model fitting software for censored data with emphasis on accelerated failure time models. We then discuss the "ANOVA" function, which can be used to compare one or more fitted models, and we describe the various visualizations that can be performed once a model has been fit. In the final sections of this Chapter, we discuss the estimation of quantiles and failure probabilities at various points for selected values of the independent variables.

For further reading on analyzing accelerated test data see Nelson (1990) or Meeker and Escobar (1998).

## THE GENERALIZED KAPLAN-MEIER ESTIMATE

The Kaplan-Meier estimator produces nonparametric estimates of failure probability distributions for a single sample of data that contains the exact time of failure, or contains data that is *right censored*. A right censored observation is one in which the failure time is only known to be greater than the time it was, for some reason, removed (*censored*) from the study or experiment. Because we consider data that may be left censored or observed in a interval and/or grouped as well, we use a generalization of the Kaplan-Meier estimate originally developed by Turnbull (1974, 1976).

### Specifying Interval Censored Data

Consider the following (artificial) table of failure times:

**Table 2:**

unit	failure	upper	Censor	censor codes
1	7	2	right	0
2	4	2	exact	1
3	5	2	exact	1
4	9	2	right	0
5	3	2	left	2
6	2	9	interval	3
7	7	12	interval	3
8	4	2	exact	1
9	11	2	right	0

First we define what we mean by the censoring types. Let  $C = (L, U)$  be a random censoring interval, and let  $T$  be the failure time, and suppose that  $C$  and  $T$  are independent (less strict assumptions are possible, see, e.g., Andersen, et al., 1993). Then an observation is an *exact failure* if the failure time  $T$  is observed so that  $T < L$ . The observation is *right censored* if the censoring time  $L$  is observed so that  $T > L$ . The observation is *interval censored* if all that is known is that  $L \leq T < U$ . Finally, the observation is left censored if all that is known is that  $0 \leq T < U$ , i.e., that the observation is

interval censored with a lower censoring time of zero.

In S-PLUS, a censoring *code* indicates the type of censoring. Censoring codes are handled quite generally allowing you to specify a set of values for each type of censoring. The default codes are: 0 means the observation is right-censored, 1 means an exact failure, 2 means a left censored observation, and 3 means an interval censored observation. To specify a censored distribution *dependent variable*, you must give both the time of failure (or censoring), and, except in exact failure (or complete) data, the censoring code. The S-PLUS function, `censor`, is used to specify the dependent variable. For the data in the Table above, you must tell the `censor` function the data type. Here the correct specification is:

```
> censor(failure, upper, censor.codes)
[1] 7+    4    5    9+    3-    [ 2,  9] [ 7, 12]
[8] 4          11+
```

When three arguments are specified to `censor`, the default censoring type is “interval”. To show the generality of the `censor` function, an alternate way of specifying the censor codes is by using the `Censor` column and stating explicitly what the codes are for each of right, left, event, and interval.

```
> cens <- censor(failure, upper, Censor, event = "exact",
right = "right", left = "left", interval = "interval")
[1] 7+    4    5    9+    3-    [ 2,  9] [ 7, 12]
[8] 4          11+
```

While this is more lengthy in this case, it is far more general allowing the user to specify a vector of codes for each of the four censoring types, event, right, left, and interval.

It is always a good idea to display the output from the `censor` function to verify that you are correctly specifying the censoring information. This is especially important because it is common practice to reverse the censoring codes for failure and right censoring, and these values must be correctly specified if the analysis results are to be meaningful. An additional check you can do is to examine the censor codes map as follows:

```
> censorCodesMap(cens)
event: exact ==> 1
right: right ==> 2
left: left ==> 3
interval: interval ==> 4
```

The internal codes 1, 2, 3 and 4 are used by the estimation routine. One other specification to `sensor` allows you to use it with other routines that require internal codes of 1 (event), 0 (right), 2 (left) and 3 (interval), i.e., `coxph`, `survreg` and `survfit`. Setting the `outCodes` argument to "0-3", results in the internal codes those routines require:

```
> cens <- sensor(failure, upper, Censor, right = "right",
+ left = "left", event = "exact", interval = "interval",
+ outCodes = "0-3")
> sensorCodesMap(cens)

event: exact ==> 1
right: right ==> 0
left: left ==> 2
interval: interval ==> 3
```

## Computing Kaplan-Meier Estimates

The `kaplanMeier` function is used to compute Kaplan-Meier estimates and Turnbull's generalization of the Kaplan-Meier estimates. For the data in the Table above, the S-PLUS statements are:

```
> kaplanMeier(sensor(failure, upper, censor.codes) ~ 1,
+ data = int.data)
```

This results in output

```
Number Observed: 9
Number Censored: 6
Confidence Type: identity

      Survival Std.Err 95% LCL 95% UCL
(-Inf, 2]    1.000  0.000  1.000  1.000
(  3,  4]    0.861  0.127  0.646  1.000
(  4,  5]    0.583  0.173  0.386  0.781
(  5,  7]    0.444  0.166  0.300  0.589
(  9, 11]    0.444  0.166  0.300  0.589
( 12, Inf)    0.000  0.000  0.000  0.000
```

In the output, each row begins with a label indicating the observation interval. The time interval is followed by the survival estimate, the standard error for the estimate and approximate confidence intervals for the estimate.

The `kapl anMei er` model computed above estimates the survival curve for a single sample. If independent variables were available in the sample, the values of all the independent variables must be identical if the results from `kapl anMei er` are to be meaningful. If an independent variable is used on the right side of the formula it is treated as a stratification variable and separate survival curves are estimated for each value of the independent variable(s).

Consider the `capaci tor2` data set distributed with S-PLUS. This data set contains four variables: 1) *days* gives the time of failure or censoring, 2) *event* gives the censoring code (1 is a failure at time *days*, while 0 is right censoring at time *days*), 3) *weights* gives the number of observations represented by that row, and 4) *voltage* gives the voltage at which the capacitor was tested (there are four distinct voltages in the data set). To analyze the failure date without regard to the test voltage, the statement

```
> kapl anMei er(censor(days, event)~1, wei ghts = wei ghts,
data=capaci tor2)
```

would be used. However, this would ignore the different test voltages. A better analysis would compute a nonparametric estimate of the failure time for each voltage. This is done with the statement

```
> km. cap <- kapl anMei er(censor(days, event) ~ vol tage
, wei ghts = wei ghts, data=capaci tor2)
```

with result

```
vol tage=20
Number Observed: 25
Number Censored: 25
[1] Not enough fai lures avail able to fit a nonparametri c
censored data model
```

```
vol tage=26
Number Observed: 50
Number Censored: 39
Confidence Type: i denti ty
```

	Survival	Std. Err	95% LCL	95% UCL
( -Inf, 12.95]	1.00	0.000	1.000	1.000
( 12.95, 28.41]	0.98	0.020	0.942	1.000
( 28.41, 63.10]	0.96	0.028	0.908	1.000
( 63.10, 136.33]	0.94	0.034	0.878	1.000
(136.33, 139.37]	0.92	0.038	0.851	0.989
(139.37, 179.02]	0.90	0.042	0.825	0.975

---

(179.02, 187.80]	0.88	0.046	0.801	0.959
(187.80, 201.28]	0.86	0.049	0.777	0.943
(201.28, 214.28]	0.84	0.052	0.755	0.925
.	.	.	.	.

For voltage=20 there are not enough observations in the sample to compute estimates. For voltage=26, voltage=29, and voltage=32, estimates are computed and displayed in a separate tables.

The Kaplan-Meier estimates of failure probabilities can also be used to compute nonparametric estimates of the quantiles. For example, the statements

```
> qkaplanMeier(km.cap, p = seq(.1, to = .9, by = .1))
```

produce the result

```
$"voltage=20":
[1] NA

$voltage=26":
 0.1    0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
139.37 271.73 Inf Inf Inf Inf Inf Inf

$voltage=29":
 0.1    0.2    0.3    0.4    0.5    0.6 0.7 0.8 0.9
45.85 55.73 91.81 108.62 164.2 257.88 Inf Inf Inf

$voltage=32":
 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
2.81 5.45 6.26 11.51 15.16 20.86 65.9 94.08 149.2
```

for the quantiles. Notice that because no failures were observed beyond 300 days, survival drops to 0.0 in the final intervals for 26 and 29 volts, resulting in quantile estimates that are infinite. The true value is, of course, finite, but is not estimable from this data.

# CENSORREG

Parametric, rather than nonparametric, estimates of the failure distributions can also be easily computed. All estimates are computed by the `sensorReg` function. Like `kaplanMeier`, `sensorReg` can handle interval and other censoring. In addition, the `sensorReg` function can handle three general families of failure distributions with logged and unlogged versions, truncated data, offsets, a “threshold” parameter, fixed coefficients, and much more.

## An Example Model

As the simplest possible example, use the defaults for most arguments in a `sensorReg` model with no covariates. Possible S-PLUS statements for the capacitor data are:

```
> sensorReg(sensor(days, event) ~ 1, weights = weights,
data=capacitor2)
```

with resulting display

```
Call :
sensorReg(formula = sensor(days, event) ~ 1, data =
capacitor2, weights = weights)
```

```
Distribution: Weibull
```

```
Coefficients:
(Intercept)
6.704817
```

```
Dispersion (scale) = 1.821207
Log-Likelihood: -372.7664
```

```
Observations: 125 Total; 71 Censored
Parameters Estimated: 2
```

As with the `kaplanMeier` function, the response is specified by the `sensor` function. Because the model formula contains no covariates, a parametric model is fit for a single sample of observations. In this case, the parametric family defaults to the *Weibull* distribution.

In the output that the location parameter for the Weibull distribution is estimated as 6.704, and the scale parameter is estimated as 1.82

As with other S-PLUS model fitting functions, the `summary` function can be used to obtain a more detailed summary of the fit. Following is the result of

calling `summary` on the fit object:

```
Call:
censorReg(formula = censor(days, event) ~ 1, data =
capacitor2, weights = weights)

Distribution: Weibull

Standardized Residuals:
      Min      Max
Uncensored 0.020 0.553
Censored   0.577 0.577

Coefficients:
      Est. Std.Err. 95% LCL 95% UCL z-value  p-value
(Intercept)  6.7    0.296   6.12   7.29   22.6 3.01e-113

Extreme value distribution: Dispersion (scale) = 1.821207
Observations: 125 Total; 71 Censored
-2*Log-Likelihood: 746
```

Specifying the Parametric Family

The parametric distribution family is specified by inputting one of the 10 distributions that are supported by `censorReg`. These are displayed in the following table 3. The `distribution` argument to `censorReg` is the quoted string in the first column along with the character string that is supplied to `censorReg` as the distribution argument.

censorReg argument	Distribution
"weibull"	Weibull
"extreme"	smallest extreme value
"lognormal"	log-normal or log-gaussian
"normal"	normal or gaussian
"loglogistic"	log-logistic
"logistic"	logistic
"logexponential"	log-exponential
"exponential"	exponential, same as extreme with sigma = 1
"lograyleigh"	log-Rayleigh
"rayleigh"	Rayleigh, same as extreme with sigma = 0.5

Table 3: Distributions supported by `censorReg`.



The following discussion describes the internal specification of the parametric distribution families as they are viewed by the estimation routines. The general user need not be concerned with this aspect of the family specification. It is included here for the user who wants or needs access to the internal routines.

Internally the distributions are defined by two quantities (following the development of standard textbooks on parametric survival analysis), the distribution of the random variable, and the link function. Let  $g(\bullet)$  denote the link function, and let

$$z = \frac{g(y) - x\beta}{\sigma}$$

be the random variable for failure time  $y$ . Here  $\sigma$  is the scale factor,  $x$  is a vector of covariates (in the simplest model  $x = 1$ , the intercept term), and  $\beta$  is a vector of coefficients. The term  $x\beta$  specifies the “location” of the estimates. Two link functions  $g(\bullet)$  are possible:

$$g(x) = x$$

the “identity” link, and

$$g(x) = \log x$$

the “log” link. Three distributions for  $z$  are available. These are the “logistic”

$$f(z) = \frac{\exp(-z)}{(1 + \exp(-z))^2}$$

the “normal” or “gaussian” distribution

$$f(z) = \frac{1}{\sqrt{2\pi}} \exp - \frac{1}{2} z^2$$

and the “smallest extreme value” distribution

$$f(z) = \exp(z - \exp(z))$$

When the log link is used with a fixed value of  $\sigma = 1$ , the smallest extreme value distribution becomes an exponential distribution. If  $\sigma = 0.5$ , this becomes the Rayleigh distribution. As indicated above, when the smallest extreme value distribution is used with the log-link, the distribution can be

made equivalent to the (two-parameter) Weibull distribution in which

$$f(z) = \frac{1}{\sigma \exp(x\beta) \left( \frac{z}{\exp(x\beta)} \right)^{\frac{1}{\sigma}-1}} \exp\left(-\left(\frac{z}{\exp(x\beta)}\right)^{\frac{1}{\sigma}}\right)$$

Here  $\theta = \frac{1}{\sigma}$  is the “shape” parameter.

In general the failure times are positive since failure at a negative time is not usually meaningful. However, when the identity link function is used, it is possible to input negative values for the survival times into `cenSorReg`. For example, a gaussian distribution takes values over the entire real line.

To fit a “gaussian” model to the capacitor2 data, you type

```
> cenSorReg(cenSor(days, event) ~ 1, data=capaci tor2,
di stri buti on="gaussi an")
```

The hazard rate is the instantaneous rate of failure. This can be computed simply as the first derivative of the failure density with respect to time. Different distributions result in different hazard rates, and thus in different models. Much time in model building can be spent in deciding upon the correct model that should be used. The plotting functions discussed below can help in making this decision.

## Accounting for Covariates

In the `cenSorReg` models above we considered only a single sample of observations from the same distribution. Typically, a survival model also includes covariate(s) to describe the distribution. Accelerated failure time models, for example, include covariates occurring in designed experiments in which the covariate is held fixed at a specified value for some observations, and the time to failure for these observations is observed. For example, in the capacitor data, four values of the covariate “voltage” were used, “voltage=20”, “voltage=26”, “voltage=29”, and “voltage=32”. Suppose that we assume that the location parameter varies linearly with the covariate, e.g., that

$$z = \frac{g(y) - \alpha_0 - \alpha_1 x}{\sigma}$$

for intercept  $\alpha_0$ . Here  $x$  is voltage. This model may be fitted using S-PLUS statements

```
> cenSorReg(cenSor(days, event) ~ vol tage, wei ghts =
wei ghts, data=capaci tor2)
```

with resulting output:

```

Call :
  censorReg(formula = censor(days, event) ~ voltage, data =
  capacitor2, weights = weights)

Distribution: Weibull

Coefficients:
  (Intercept)      voltage 
    24.14083   -0.6403586

Dispersion (scale) = 1.203945
Log-likelihood: -316.4589

Observations: 125 Total; 71 Censored
Parameters Estimated: 3

```

In the above model the location parameter is obtained by “regression” on the voltage. This requires a *linear* relationship of the hazard rate on voltage. Assuming that the relationship is not linear, a more general model fits

$$z = \frac{g(y) - \alpha_0 - \alpha_i}{\sigma}$$

In this model,  $i$  indexes the different voltages, and the location parameter is allowed to vary in an arbitrary manner with voltage. Fitting this model is accomplished simply as

```

> censorReg(censor(days, event) ~ factor(voltage), weights
= weights, data=capacitor2)

```

Alternatively, supposing that the scale parameters are different for different values of the covariate, a model

$$z = \frac{g(y) - \alpha_0 - \alpha_i}{\sigma_i}$$

can be fit using S-PLUS statements

```

> censorReg(censor(days, event) ~ strata(voltage), weights
= weights, data=capacitor2)

```

In all but the last case, an object of class “`censorReg`” is produced. In the last example when the `strata` function is used to create a stratified fit, an object of class “`censorRegList`” is produced. This object contains a list of class “`censorReg`” objects.

The `anova` function is used to compare the models described above. This is discussed in more detail below.

## Truncation Distributions

Aside from the distributions above, it is also possible to specify a different truncation distribution for each observation. Consider the following table of failure times.

**Table 4:**

Unit	failure	upper	censor	censor.code	tlower	tupper	trunc codes
1	7	2	right	0	3	2	2
2	4	2	exact	1	0	2	1
3	5	2	exact	1	0	2	1
4	9	2	right	0	3	2	2
5	4	2	left	2	9	2	0
6	5	9	interval	3	3	20	3
7	7	12	interval	3	3	20	3
8	4	2	exact	1	0	2	1
9	11	2	right	0	3	2	2

In truncated data the item being tested is not observed over the entire positive axis. Instead, observation of the item is made over a known interval that is a subset of the time period in which the observation could fail. Thus, if there is left truncation, the items under test may be manufactured, used for a time, and then placed on test. Although the time to failure is scored as the time since manufacture, items that fail prior to being placed on test are not scored. Let  $t = 0$  be the time of manufacture, and suppose that testing is not begun until  $t = \theta$ . Then if  $F(\theta)$  is the cumulative distribution of the failure time when observation starts at time zero, then the cumulative distribution of the truncated failure times is given by

$$F(t|\theta) = \frac{F(t)}{1 - F(\theta)}$$

Similarly, in right truncation, observation of failure or censoring is only made

until  $t = \theta$  so that observations that fail or are censored after time  $\theta$  cannot be observed (or are thrown out). Finally, in interval truncation, observation is made over a fixed interval  $(\theta_1, \theta_2)$ , and observations that fail or are censored outside of the interval are not considered.

Truncation distributions can easily be fit using the `tensorReg` function. For example, to obtain a “gaussian” fit to the data above, one would use:

```
> tmp <- tensorReg(tensor(failure, upper, cens) ~ 1,
  data=table4, truncation = tensor(tlower, tupper, tcode),
  distribution = "lognormal")
```

which results in output:

```
Call:
tensorReg(tensor(failure, upper, cens) ~ 1, data = table4,
  truncation = tensor(tlower, tupper, trunc.codes),
  distribution = "lognormal")

Distribution: Lognormal

Coefficients:
(Intercept)
      1.920974

Dispersion (scale) = 0.9211897
Log-likelihood: -12.49965

Observations: 9 Total; 6 Censored
Parameters Estimated: 2
```

Because the log-likelihood is more complex (numerically) when truncation distributions are used, it is important to verify convergence. Here convergence is verified by the near zero values of the first derivatives of the log-likelihood. The above model was temporarily save in `tmp`, so we can extract the derivatives as follows:

```
> tmp$first.deriv
      (Intercept)      scale
-6.594777e-010 -4.993228e-009
```

## Threshold Parameter

Truncation distributions modify the fitted distribution by considering failure in a smaller region of the positive real line. A distribution with a threshold parameter also modifies the failure distribution, but in a slightly different way. The idea of the threshold parameter is that test items cannot fail for a period of time after testing begins. Thus, although testing begins at time zero, no tested item will fail for some fixed period  $\gamma$  after time zero. Thus, the failure distribution is given by  $F(t|\gamma) = F(t - \gamma)$ . The net effect of the threshold parameter is to shift the failure distribution to the right by a fixed amount.

Maximum likelihood estimation of  $\gamma$  is not easily accomplished. There is some discussion of this in Meeker and Escobar (1998, pp. 224 - 231). You can either compute the value of  $\gamma$  yourself and enter it as input to the `sensorReg` function, or `sensorReg` can be asked to estimate  $\gamma$  in two different ways. The first is to simply decrease the smallest value by 10%. The second works only for log distributions and computes a value for  $\gamma$  which optimally linearizes a qqplot of the Kaplan-Meier estimate of survival and the (censored) observations. By default,  $\gamma = 0$ . Once computed,  $\gamma$  is carried along with the `sensorReg` object.

For the example in the table above we can set the *threshold* parameter to equal two as follows:

```
> sensorReg(sensor(failure, upper, cens) ~ 1, data =
table4, truncation = sensor(tlower, tupper, tcode),
distribution = "lognormal", threshold = 2)
```

This yields output:

Call:

```
sensorReg(formula = sensor(failure, upper, censor.codes) ~
1, data = table4, truncation = sensor(tlower, tupper,
trunc.codes), distribution = "lognormal", threshold = 2)
```

Distribution: Lognormal

Coefficients:

```
(Intercept)
1.664897
```

Dispersion (scale) = 1.38711

Log-Likelihood: -12.23809

Observations: 9 Total; 6 Censored

Parameters Estimated: 2

Threshold Parameter: 2

Notice that the coefficient estimates have dramatically changed.

## Offsets

Offsets are also used to change the distribution of the failure time variable. Let  $\omega$  denote the offset and let  $y$  denote the failure time. When offsets are used, the transformed failure time becomes

$$z = \frac{g(y) - \omega - x\beta}{\sigma}$$

where the offset  $\omega$  is a known and fixed value.

A typical use of offsets is in likelihood ratio tests. Suppose that  $x_1\hat{\beta}_1 + x_2\hat{\beta}_2$  optimizes the likelihood when covariates  $x_1$  and  $x_2$  are included in the model. Then a likelihood ratio test of  $H_0: \hat{\beta}_1 = \kappa$  is obtained by setting  $\bar{\omega} = x_1\kappa$  and comparing the optimized value of the likelihood of a model  $\bar{\omega} + x_2\hat{\beta}_2$  with the optimized likelihood for model  $x_1\hat{\beta}_1 + x_2\hat{\beta}_2$ .

We illustrate using the capacitor2 failure data discussed above. When “voltage” is included in the model the output is:

```
Call :
  censorReg(formula = censor(days, event) ~ voltage, data =
  capacitor2, weights = weights)

Distribution: Weibull

Coefficients:
  (Intercept)      voltage 
    24.14083   -0.6403586 

Dispersion (scale) = 1.203945
Log-likelihood: -316.4589

Observations: 125 Total; 71 Censored
Parameters Estimated: 3
```

A likelihood ratio test that the voltage coefficient is fixed at -0.5 is obtained by fitting a second model with offset specified to fix the parameter estimate of voltage.

```
> censorReg(censor(days, event) ~ offset(-0.5*vol tage),  
weights = wei ghts, data=capaci tor2)
```

which yields output

```
Call :  
censorReg(formula = censor(days, event) ~ offset(-0.5 *  
vol tage), data = capaci tor2, wei ghts = wei ghts)
```

```
Di stri buti on: Wei bul l
```

```
Coeffi ci ents:
```

```
(Intercept)
```

```
19.94567
```

```
Di spersi on (scal e) = 1.090527
```

```
Log-l i kel i hood: -1129.826
```

```
Observations: 125 Total; 71 Censored
```

```
Parameters Estimated: 2
```

```
Offset has been speci fi ed
```

Computing the likelihood ratio test from the above two fits by hand we get

$$\text{LRT} = -2 * (-1129.8 + 316.5) = 1626.6$$

which is compared with a chi-squared distribution with one degree of freedom. Clearly, this is a significant result.

## Fixing parameters

It is also possible to simply fix parameters in the model. Most often this will be the scale parameter, but it is possible to fix any parameter. For example, in the capacitor example we may fix the voltage coefficient to be -0.5 using

```
> censorReg(censor(days, event) ~ vol tage, data=capaci tor2,  
weights = wei ghts, fi xed=list(vol tage=-0.5))
```

which gives result:

```
Di stri buti on: Wei bul l
```

```
Coeffi ci ents:
```

```
(Intercept)
```

```
19.94567
```

```
Di spersi on (scal e) = 1.090527
```

```
Log-l i kel i hood: -1129.826
```



Observations: 125 Total; 71 Censored

Parameters Estimated: 2

Comparing this with the results in which offset is set, we see that the effect of fixing voltage to be -0.5 is the same as specifying the offset as  $-0.5 \times \text{voltage}$ .

## FITTING MODELS: ANOVA

The `anova` function is used to compare models. If a single object is input to `anova`, then one term at a time is added to the model starting from the smallest possible model (usually the intercept-only model) until the model contained in the object is obtained. As an example, consider the following model:

```
> fi t <- censorReg(censor(days, event) ~ vol tage +
  vol tage^2, weights = weights, data = capaci tor2)
```

Applying the `anova` function to `fi t` as follows

```
> anova(fi t, test = "Chi ")
```

produces

Likelihood Ratio Test Table

Weibull model

Response: censor(days, event)

Terms added sequentially (first to last)

	N. Params	-2*LogLik	Df	LRT	Pr(Chi)
NULL	2	745.5327			
vol tage	3	632.9178	1	112.6149	0.0000000
I (vol tage^2)	4	632.8494	1	0.0684	0.7937407

It is suggested by the display that the location parameter of the distribution depends on voltage only linearly. The quadratic term is unimportant. We'll verify this with other models and graphically below.

When two or more class "`censorReg`" or class "`censorRegList`" objects are input into the `anova` function, the models are compared with likelihood ratio tests. Suppose we are interested in testing whether the model for the capacitor data should be

$$z = \frac{g(y) - x\beta}{\sigma}$$

where  $x$  is voltage. More general models (in the sense of having more parameters) are

$$z = \frac{g(y) - \alpha_i}{\sigma}$$

for voltage  $i$ , or

$$z_i = \frac{g(y) - \alpha_i}{\sigma_i}$$

These three models plus an intercept-only model can be generated in S-Plus using the following statements:

```
> fi t0 <- censorReg(censor(days, event) ~ 1, weights =
weights, data=capaci tor2)

> fi t1 <- censorReg(censor(days, event) ~ vol tage, weights =
weights, data=capaci tor2)

> fi t2 <- censorReg(censor(days, event) ~ factor(vol tage),
weights = wei ghts, data=capaci tor2)

> fi t3 <- censorReg(censor(days, event) ~ strata(vol tage),
weights = wei ghts, data=capaci tor2)
```

The models are then compared using the `anova` function as follows:

```
> anova(fi t0, fi t1, fi t2, fi t3, test="Chi sq")
```

which yields the display:

Li kel i hood Ra ti o Test(s)

Response: censor(days, event)

	Terms	N. Params	-2*LogLi k	Test	Df	LRT	Pr(Chi )
1		1	2	745.53			
2	vol tage	3	632.92	+ vol tage	1	112.615	0.0000
3	factor(vol tage)	5	632.37	2 vs. 3	2	0.547	0.7605
4	strata(vol tage)	6	630.40	3 vs. 4	1	1.973	0.1601

The evidence is now quite strong that we can't do any better than the model which relates the location parameter of the distribution to a linear regression (single parameter) model in voltage. We can verify this by looking at graphics

## FITTING MODELS: THE PLOT METHOD FOR CENSORREG

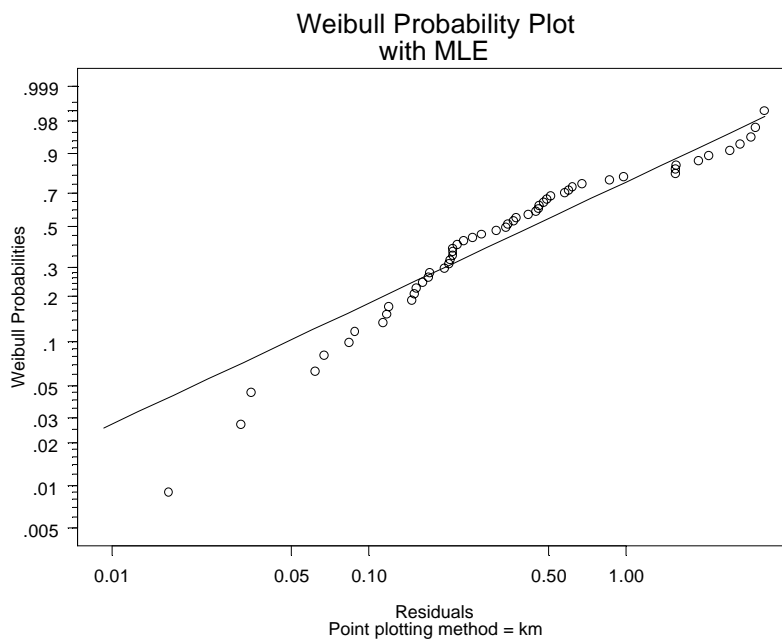
The plot method for objects of class “`ensorReg`” generates 4 to 6 plots depending on the type of fit. You can generate all possible plots for a “`ensorReg`” fit object by simply using the plot function as follows:

```
> plot(fi t1)
```

The first three plots resulting from the above call are equivalent to those produced for fit objects of class “`lm`” or “`glm`” so they won’t be discussed further here.

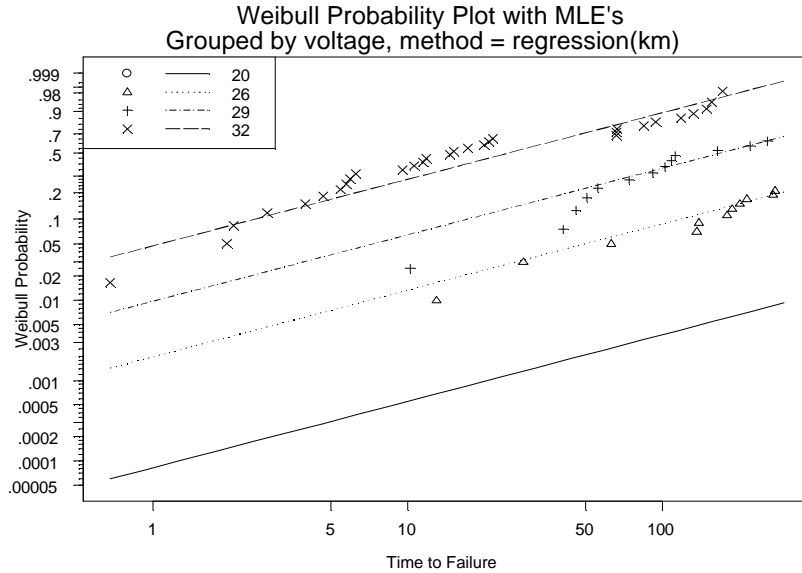
The last four are different and are presented in figure 13.1 through figure 12.4.

Figure 13.1 displays a probability plot of the standardized residuals. The



**Figure 13.1:** *Probability plot of standardized residuals with maximum likelihood estimate.*

standardization of the residuals are described in Meeker and Escobar (1998) and are referred to by them as “censored Cox-Snell” residuals. A maximum likelihood estimate of a null model (intercept only) is displayed in the plot along with the residuals for diagnostic purposes.

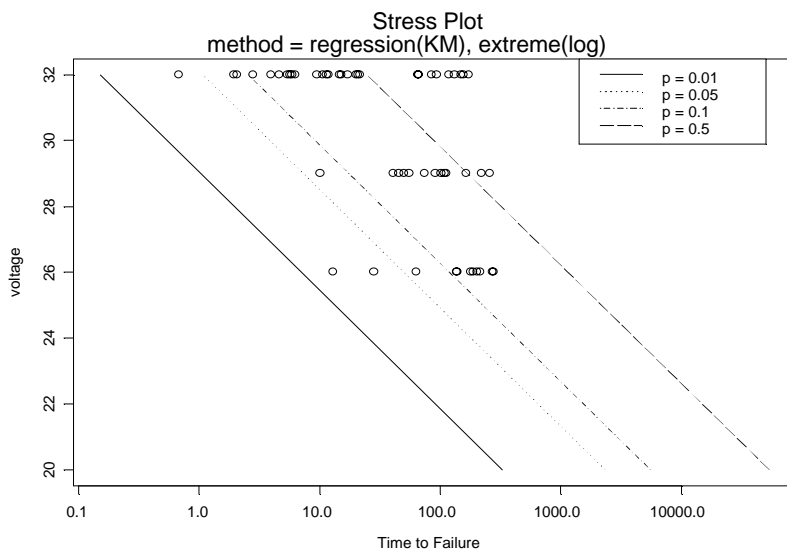


**Figure 13.2:** *Probability plot of the fit with maximum likelihood estimates.*

Figure 13.2 displays a probability plot of the fitted model along with the non-censored observations. Each line and each set of points corresponds to the fit and non-censored observations for a different value of the covariate. This plot gives a good assessment of the fit. However, it is currently only available for single covariate models. The `tensorReg` function is not constrained to single covariates, but this plotting function is. You can access this function directly by calling `probplot.tensorReg`. See the help file for `probplot.tensorReg` for more details.

Figure 13.3 displays what engineers refer to as a *stress* plot. It plots the non-censored observations and equi-probability lines for the predictor variable (the stressor) versus failure times. It is quite clear from the graph that as voltage (stress) decreases, failure times increase. This plot is also constrained

to single covariate regression models. For more details see the help file for [stressplot.censorReg](#).

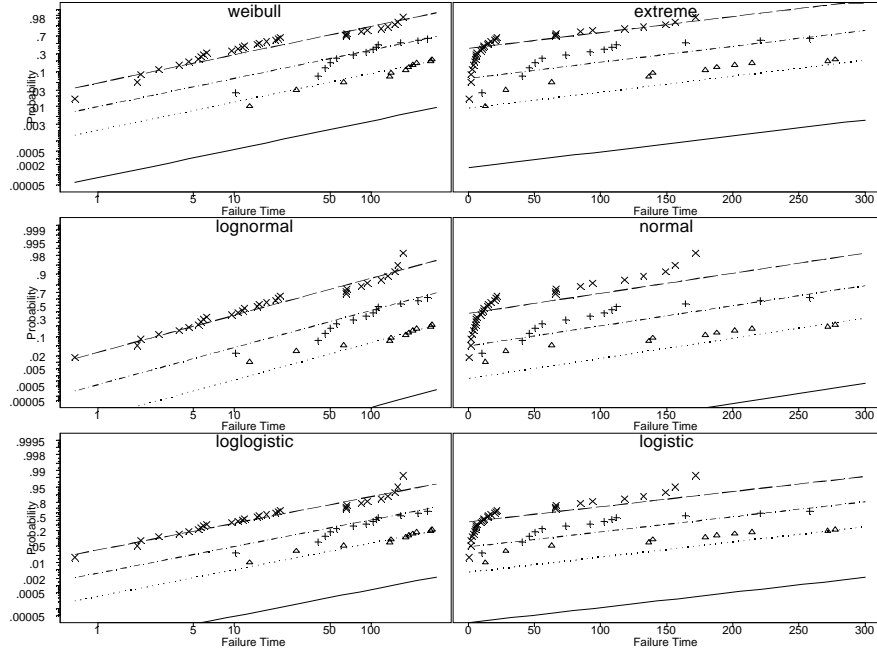


**Figure 13.3:** *Stress plot of the fit.*

The final diagnostic plot, also for a fit with a single covariate, is displayed in figure 13.4. This is the same plot as figure 13.2 but repeated for six distributions. The distributions are the weibull, the lognormal and loglogistic coupled with their non-logged counterparts. This plot is provided primarily for distribution assessment. It's quite clear from figure 13.4 that a non-logged distribution does not fit the data well. Exactly which logged distribution fits best is not so clear. For more information on this plot function see the help file for [probplot6.censorReg](#).

As mentioned above the three plotting functions [probplot.censorReg](#), [stressplot.censorReg](#), and [probplot6.censorReg](#) are called by the plot method for a "[censorReg](#)" object. These functions, however, were designed to be called directly and provide more capabilities than are available through the general plot method. One primary example of this is the [method](#) argument to each of these plotting functions which allows the

plotted points to be computed based upon some alternative model. This



**Figure 13.4:** *Six-distribution plot of the fit.*

argument defaults to the "KM", or Kaplan-Meier estimates, but four other sets of estimates are possible. These are 1) "one", or null (intercept only) model in which case

$$z = \frac{g(y) - \mu}{\sigma}$$

for location parameter  $\mu$ , 2) "regressi on" model which allows

$$z = \frac{g(y) - x\beta}{\sigma}$$

for covariate  $x$ , 3) "factor" models which uses

$$z = \frac{g(y) - \alpha_i}{\sigma}$$

for covariate values  $i$  to compute separate locations for each value of the covariate, and finally, 4) "separate" model is the most general single variable parametric model which allows separate location and scale parameter estimate for

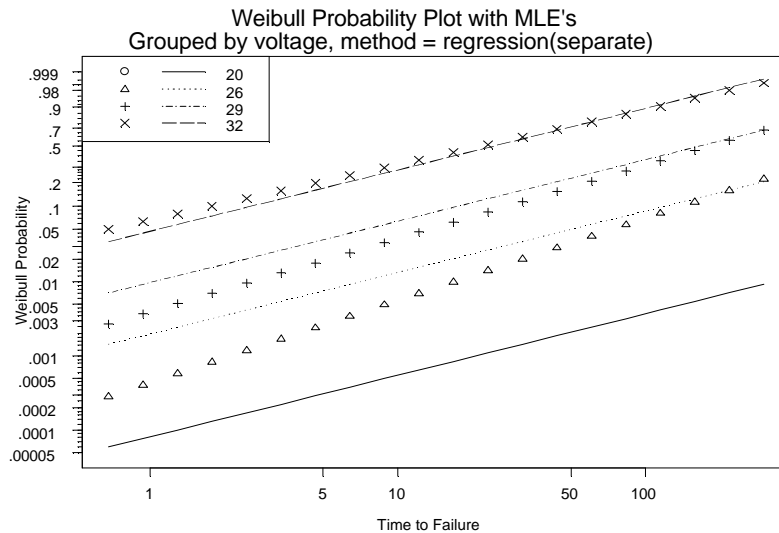
each value of the covariate:

$$z_i = \frac{g(y) - \alpha_i}{\sigma_i}$$

For our example, comparing the regression fit with the more general "separate" fit in the probability plot is accomplished using the statement

```
> probplot(fit1, method="separate", add.legend=T,
  legend.loc = "auto")
```

which results in the following plot.



**Figure 13.5:** *Probability plot for comparing models.*

The plotted points in figure 13.5 are obtained from the "separate" model and show some deviation from the "regression" model. However this is not statistically significant as we saw previously when we compared the models using a likelihood ratio test. You can also add confidence intervals to the plot for each maximum likelihood estimate to get a feel for the variability of the estimated distribution(s).



## COMPUTING PROBABILITIES AND QUANTILES

The `predict` method for “`ensorReg`” objects computes predictions from a fitted model on either probability or response scales at designated quantiles or probabilities, respectively, for specified covariate values. For example, suppose you want to estimate the time to 10%, 50% and 90% failure from our regression model for the `capaci tor2` data for values of voltage at 16, 20, and 24. The the call to the `predict` function is

```
> predict(fi t1, newdata = data.frame(vol tage = c(16,
20, 24)))
```

with resulting display:

```
$"vol tage=16":
```

	Estimate	Std.Err	95% LCL	95% UCL
0.1	72097.22	1.028782	9598.82	541525.8
0.5	696503.03	1.133190	75570.05	6419427.9
0.9	2955616.38	1.217862	271644.96	32158403.5

```
$"vol tage=20":
```

	Estimate	Std.Err	95% LCL	95% UCL
0.1	5565.468	0.7211182	1354.206	22872.76
0.5	53765.809	0.8136006	10913.602	264877.00
0.9	228155.656	0.8986737	39199.343	1327956.02

```
$"vol tage=24":
```

	Estimate	Std.Err	95% LCL	95% UCL
0.1	429.6203	0.4384364	181.9228	1014.571
0.5	4150.3943	0.5003670	1556.5971	11066.302
0.9	17612.2327	0.5853507	5591.9462	55470.980

Operating the capacitor at 16 volts increases its life span by about 170 times compared to operating at 24 volts. The probability values (proportion failed) are 0.1, 0.5, and 0.9 by default when calling the `predict` function. That can be modified by specifying the `p` argument. For example to compute the 10%, 20% and 30% failure times you would enter

```
> predict(fi t1, p = c(.1, .2, .3), newdata =
data.frame(vol tage = c(16, 20, 24)))
```

Alternatively, to predict proportion failed or failure rates for given quantiles of the failure time distribution, you specify `type = "probability"` as an argument to `predict`. Let's compute the failure rates for the same set of voltage values at 1000, 2000 and 3000 days.

```
> predict(fit1, q = c(1000, 2000, 3000), type = "prob",  
newdata = data.frame(voltage = c(16, 20, 24)))
```

```
$"voltage=16":
```

	Estimate	Std. Err	95% LCL	95% UCL
1000	0.003011831	0.7981976	0.0006315958	0.01423447
2000	0.005350046	0.7977207	0.0011250641	0.02504342
3000	0.007484339	0.7997419	0.0015703341	0.03489259

```
$"voltage=20":
```

	Estimate	Std. Err	95% LCL	95% UCL
1000	0.02500204	0.5845648	0.008088394	0.07462304
2000	0.04403085	0.5949867	0.014147239	0.12879193
3000	0.06111373	0.6058481	0.019466567	0.17587955

```
$"voltage=24":
```

	Estimate	Std. Err	95% LCL	95% UCL
1000	0.1914712	0.4138868	0.09520448	0.3476748
2000	0.3147595	0.4679518	0.15510243	0.5347458
3000	0.4110080	0.5191918	0.20142736	0.6587655

The difference is again dramatic when comparing 16 and 24 volts. After 1000 days you expect only about 3 out of 1000 capacitors to fail when operated at 16 volts compared to 19 out of 100 when operated at 24 volts.

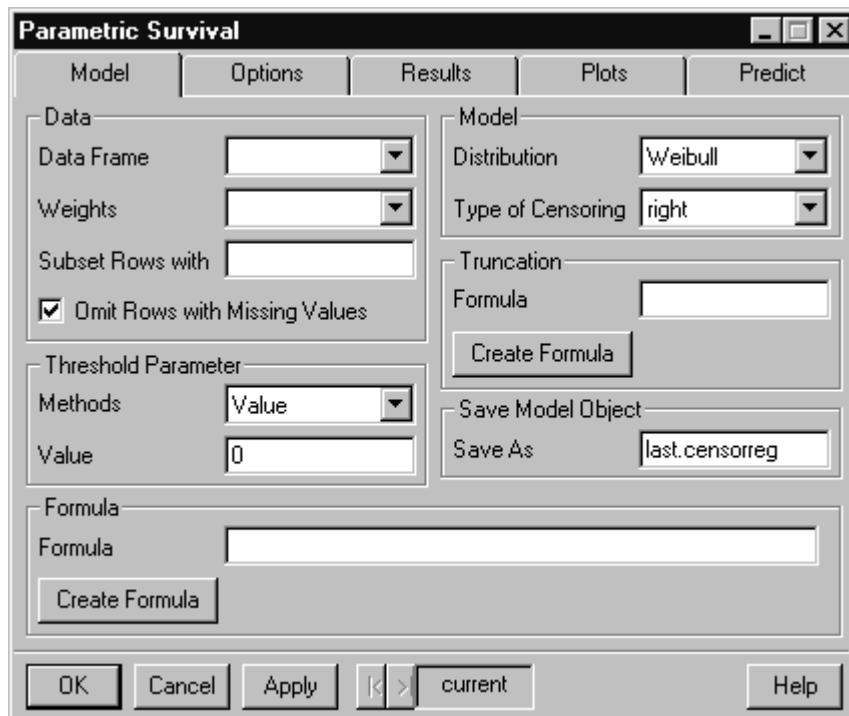
Additional arguments to `predict` allow you to specify the confidence level (referred to as *coverage* by the function) of the confidence intervals and whether or not you want to print the standard errors and confidence intervals.

# PARAMETRIC SURVIVAL

This dialog fits a regression model to survival or more generally *censored* data. See also chapter 22, Overview of Survival Analysis, and chapter 25, Parametric Regression in Survival Models, in the *Guide to Statistics*.

To perform parametric survival modeling:

Choose **Statistics ► Survival ► Parametric Survival** from the main menu. The dialog shown below appears.



**Figure 13.6:** *The Parametric Survival dialog, Model page.*

## Model Page

### Data

#### Data Frame

Select a data frame.

► **Tip ...**

*You can type into the Data Frame edit box any expression which evaluates to a data frame.*

#### Weights

Enter the column that specifies weights to be applied to all observations used in the regression. To weight all rows equally, leave this blank.

#### Subset Rows with

Enter an S-PLUS expression which identifies the rows to use in the analysis. To use all the rows in the data frame, leave this field blank. The expression must evaluate to a vector of logical values (TRUE values are used, FALSE values are dropped), or a vector of indices identifying the numbers of the rows to use.

Examples:

<code>species == "Bear"</code>	only Bears are used
<code>voltage != 20</code>	all voltages are used except 20
<code>1:20</code>	only the first 20 rows of the data are used.
<code>Age &gt;= 13 &amp; Age &lt; 20</code>	only teenagers are used.

For more information on constructing logical expressions, see the *S-PLUS Programmer's Guide*.

#### Omit Rows with Missing Values

Check this box to omit from the analysis any rows in the data frame that contain missing values for any of the variables in the model.

If this box is not checked, S-PLUS will report an error and halt the routine if any row is found to have a missing value in any of the terms in the model.

### Formula

#### Formula

Enter a formula specifying the desired model.

Examples:

```
censor(days, event) ~vol tage
```

### Create Formula

Click this to open a formula builder dialog used to construct a formula specifying the desired model. See the chapter on Building Formulas for more information.

## Model

### Distribution

Select the assumed distribution for the transformed response variable.

### Type of Censoring

Specify the type of censoring: right, left, counting, interval.

## Truncation

### Formula

Enter a formula specifying the truncation model.

Examples:

```
censor(days, event)
```

### Create Formula

Click this to open a formula builder dialog used to construct a formula specifying the truncation model. See the chapter on Building Formulas for more information.

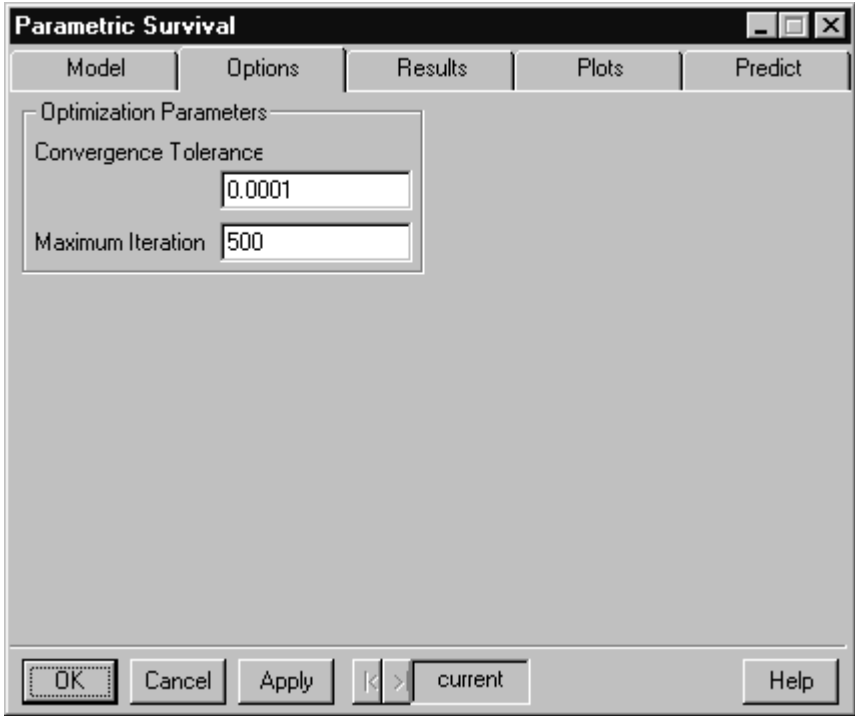
## Save Model Object

### Save As

Enter the name for the object in which to save the results of the analysis. If an object with this name already exists, its contents will be overwritten.

This must be a valid S-PLUS object name—any combination of alphanumeric characters that starts with an alpha character is allowed. The only non-alphanumeric character allowed is the period ".". Names are case-sensitive, so X and x are different names. Where appropriate, Save As defaults to a name that starts with "last". For example, "last.censorreg" is the most

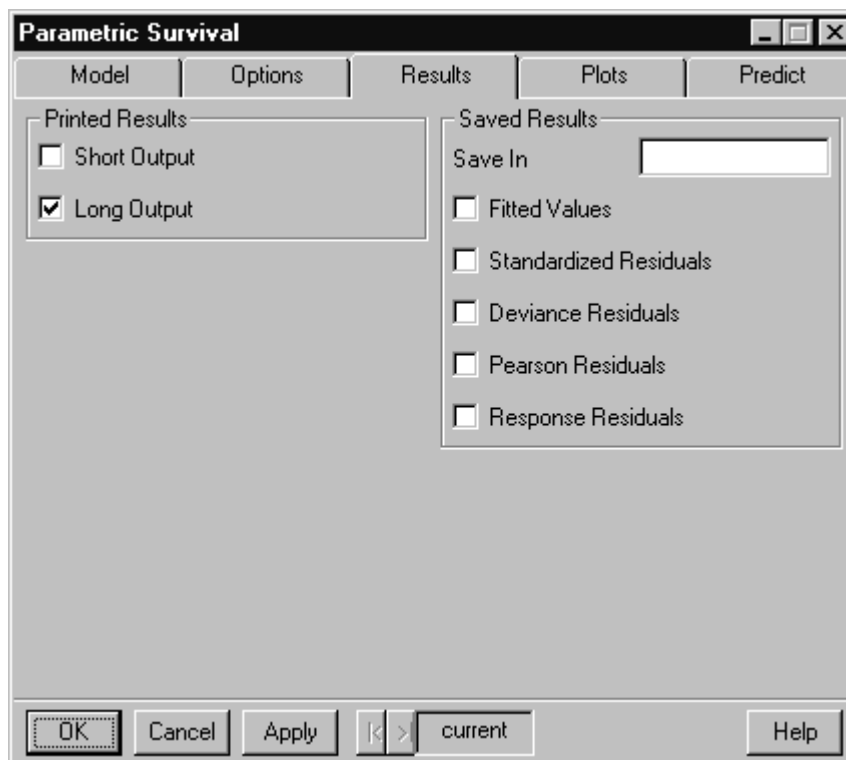
recent parametric survival model fit.



**Figure 13.7:** *The Parametric Survival dialog, Options page.*

<b>Options Page</b>	Use the Options page to define optimization parameters for model computations.
<b>Optimization Parameters</b>	<b>Convergence Tolerance</b> Enter a number specifying the convergence tolerance. Iteration will continue until the relative change in deviance is less than this number. <b>Maximum Iteration</b> Enter a number specifying the maximum number of iterations. If convergence has not been reached after this number of iterations, the

procedure will stop.



**Figure 13.8:** *The Parametric Survival dialog, Results page.*

## Results Page

### Printed Results      Short Output

Check this to print a summary of the model results in the designated output window. This includes estimates of the coefficients, dispersion (scale), degrees of freedom, and  $-2 \times \text{loglikelihood}$ .

### Long Output

Check this to print a long summary of the model results in the designated output window. This includes summary statistics for the deviance residuals,

standard errors and z-values for the coefficients, number of iterations, and correlation of coefficients.

## **Saved Results**

### **Save In**

Enter the name of an S-PLUS data frame in which a part, such as fitted values and residuals, of the analysis is saved. If an object with the name you enter does not already exist (in database 1), then it will be created. If you enter the name of a data frame that already exists (in database 1) and this data frame has the same number of rows as the number of observations used in the model fit, then the saved values are appended to this data frame. This allows you to keep fitted values from a model with the original data or to keep the residuals from a number of different models for the same data in one data frame. If you give the name of an existing S-PLUS object that is not a data frame or is not the appropriate size, then a warning is issued and a modified name is used.

### **Fitted Values**

Check this to save the fitted values from the model in the object specified in Save In.

### **Standardized Residuals**

Check this to save the standardized residuals.

### **Deviance Residuals**

Check this to save the deviance residuals. The sum of squares of these added up to the deviance.

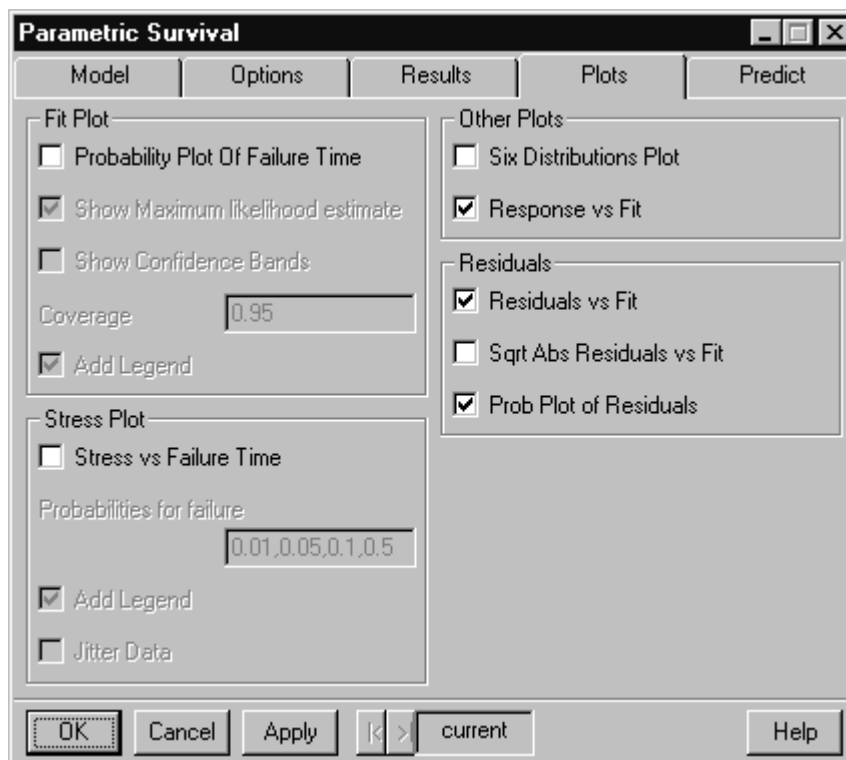
### **Pearson Residuals**

Check this to save the pearson residuals. These are standardized residuals on the scale of the response.

### **Response Residuals**

Check this to save the response residuals





**Figure 13.9:** *The Parametric Survival dialog, Plots page.*

## Plots Page

### Fit Plot

#### Probability Plot Of Failure Time

Check this to plot failure probability versus failure time in models with one or fewer covariates or stratification variables.

#### Show Maximum likelihood estimate

Check this to plot the maximum likelihood estimate of the failure probability on the fit plot.

**Show Confidence Bands**

Check this to plot the confidence bands for the maximum likelihood estimate of the failure probability.

**Coverage**

Type the confidence level for the maximum likelihood estimation.

**Stress Plot****Stress vs. Failure Time**

Check this to plot stress versus failure time.

**Probability for failure**

A vector of probabilities for which quantiles are to be computed. The quantiles at each covariate value are plotted and quantiles with the same probability are connected with a line.

**Other Plots****Six Distributions Plot**

Check this to plot six distributions of the fitted model.

**Response vs. Fit**

Check this to plot the response variable versus the fitted values. The line  $y = x$  is also drawn on the graph.

**Residuals****Residuals vs. Fit**

Check this to plot the deviance residuals versus the fitted values.

**Sqrt Abs Residuals vs Fit**

Check this to plot the square root of the absolute values of the deviance residuals versus the fitted values. This plot is useful for checking for the constant variance assumption of the model.

## Probability Plot of Residuals

Check this to create a probability plot of the standardized residuals.

**Parametric Survival**

Model Options Results Plots Predict

New Data

☐ Predict Probabilities

☐ Predict Response

Predict Probabilities

At Response Value

Confidence Level

Predict Response

At Probability Values

Confidence Level

Save Results

Save In

☒ Print Results

OK Cancel Apply |< >| current Help

**Figure 13.10:** *The Parametric Survival dialog, Predict page.*

## Predict Page

### New Data

Enter the name of a data frame to use for computing predictions. It must contain the same names as the terms in the right side of the formula for the model. If omitted, the original data are used for computing predictions.

### Predict Probabilities

Check this to predict probability of failure.

**Predict Response**

Check this to predict the response based upon the fitted probability distribution.

**Predict Probabilities****At Response Values**

A vector of response values used to predict probabilities. Enabled only while 'Predict Probabilities' is checked.

**Confidence Level**

Type confidence level used for confidence intervals of predicted probabilities.

**Predict Response At Probabilities**

A vector of probabilities used to predict the response. Enabled only while 'Predict Response' is checked.

**Confidence Level**

Type confidence level used for confidence intervals of predicted response values.

**Save****Save In**

Enter the name of an S-PLUS list in which predictions and standard errors are to be saved.

**Print Results**

Check this to print out the result of prediction.

**Related S-PLUS language functions for Parametric Survival:**

```
sensorReg, print.sensorReg, plot.sensorReg,  
summary.sensorReg, residuals.sensorReg,  
sensorReg.control, sensorReg.fit,  
sensorReg.distributions, anova.sensorReg, pftdist,  
qftdist.
```

**Other related S-PLUS language functions:**

```
formula, lm, solve, censor
```

# NEW GUI TOOLKIT FUNCTIONS

# 14

The S-PLUS GUI toolkit is a set of S-PLUS functions that enables communications between S-PLUS applications and Windows. For S-PLUS Version 4.5, the GUI toolkit has been expanded. Chapter 11 of the S-PLUS *Programmer's Guide*, Programming the User Interface using S-Plus, includes a thorough discussion of the GUI toolkit as it existed in S-PLUS 4.0; at least a casual reading of that chapter is a helpful prerequisite to the present chapter.

This chapter describes how to use the new GUI toolkit functions to either query the GUI for existing settings, or to allow S-PLUS functions to alter the settings in the GUI.

## guiSetOption

Use the `guiSetOption` function to set options available in the GUI under the Options menu. For example, to disable Tool Tips in dialogs, you would use `guiSetOption` as follows:

```
> guiSetOption("Tool TipsForDialogs", "F")
```

## guiGetOption

Use the `guiGetOption` function to obtain the current value of any option available in the GUI under the Options menu. For example, to get the current Trellis background color, use `guiGetOption` as follows:

```
> guiGetOption("BackColorTrellis")  
[1] "Lt Gray"
```

## guiSetRowSelections

Use the `guiSetRowSelections` function to specify one or more rows of a data set as "selected"; that is, they appear highlighted in a Data window view, and plotted symbols appear highlighted in a Graphsheet window. This

selection can be done interactively in the GUI; this function permits the same behavior programmatically. This is useful, for example, if you want to highlight known outliers in a data set.

## guiGetRowSelections

Use the `guiGetRowSelections` function to obtain a list of rows in the current data set that are selected.

## guiGetRowSelectionExpr

Use the `guiGetRowSelectionExpr` function to obtain an S-PLUS expression for the set of rows currently selected in a GraphSheet or Data Window. For example, consider the Data Window shown in Figure 14.1.

fuel.frame		
		1
		Weight
46	Mazda 929 V6	3480.00
47	Nissan Maxima V6	3200.00
48	Oldsmobile Cutlass Ciera 4	2765.00
49	Oldsmobile Cutlass Supreme V6	3220.00
50	Toyota Cressida 6	3480.00
51	Buick Le Sabre V6	3325.00
52	Chevrolet Caprice V8	3855.00
53	Ford LTD Crown Victoria V8	3850.00

**Figure 14.1:** *Data Window with two rows highlighted in fuel.frame.*

---

Rows 46 and 51 of the `fuel.frame` data set are selected. To store this information for future use, you can use `guiGetRowSelectionExpr` as follows:

```
> guiGetRowSelectionExpr("fuel.frame")  
[1] "46, 51"
```

You can select those same rows in a later session using `guiSetRowSelection`:

```
> guiSetRowSelection("fuel.frame", "46, 51")
```

## guiPrintClass

Use the `guiPrintClass` function to obtain a list of properties for any GUI class, and for each property, a list of acceptable values. You can use the results of this function to help construct calls to `guiCreate` and `guiModify`. For example, suppose you wanted to make a line plot. You could call `guiPrintClass` on the class `"LinePlot"` and see what properties such a plot contains, then construct a call to `guiCreate` to build the plot you wanted, as follows:

```
> guiPrintClass("LinePlot")  
CLASS:    LinePlot  
ARGUMENTS:  
  Name  
    Prompt:  
    Default:  
  DataSet  
    Prompt:    Data Set  
    Default:    ""  
  xColumn  
    Prompt:    x Column(s)  
    Default:    ""  
  yColumn  
    Prompt:    y Column(s)  
    Default:    ""  
  ...  
  LineStyle  
    Prompt:    Style  
    Default:    "Solid"  
    Option List: [ None, Solid, Dots, Dot Dash, Short Dash,  
                  Long Dash, Dot Dot Dash, Alt Dash, Med Dash, Tiny Dash ]  
  LineColor  
    Prompt:    Color  
    Default:    "Cyan"  
    Option List: [ Black, Blue, Green, Cyan, Red, Magenta,  
                  Brown, Lt Gray, Dark Gray, Lt Blue, Lt Green, Lt Cyan,
```

```

      Lt Red, Lt Magenta, Yellow, Bright White, Transparent,
      User1, User2, User3, User4, User5, User6, User7, User8,
      User9, User10, User11, User12, User13, User14, User15,
      User16 ]
LineWeight
  Prompt:      Weight
  Default:     "1"
  Option List: [ Hairline, 1/4, 1/3, 1/2, 1, 2, 3, 4, 5,
                6, 8, 10, 12 ]
  ...
> gui Create("LinePlot", DataSet="fuel.frame", xColumn="Weight",
+           yColumn="Mileage", LineStyle="Alt Dash", LineColor="Magenta")

```

S-PLUS provides default values for most unspecified properties; thus, the plot produced by the above command shows cyan open circles at each data point. The default values for plot colors, line styles, and other basic characteristics are set in Options/Graph Styles. Other defaults can be modified by saving the object as a default.

## guiPlot

Use the `guiPlot` function as a convenient way to create editable graphics from S-PLUS functions. Unlike `guiCreate` and `guiModify`, which can be used to create graphics but are also used to create other GUI objects, `guiPlot` is used exclusively to create graphics. It therefore has a simpler and more intuitive syntax.

For example, suppose you want to create two line plots on the same graph in a new graph sheet, and store the data within the graph sheet. The following calls do exactly that:

```

> x <- 1:30
> guiPlot("Line", DataSetValues=data.frame(x, cos(x),
      sin(x)))
[1] "GS2"

```

Suppose you want to create a Trellis graph with two conditional variables. You can do this with `guiPlot` as follows:

```

> guiPlot("Loess", DataSetValues=environmental,
      NumConditioningVars=2)
[1] "GS3"

```

## guiRefreshMemory

Use the `guiRefreshMemory` to remove unneeded objects from memory; you



---

can optionally restore the object's summary data after clearing the entire object from memory.

## guiExecuteBuiltIn

Use the `guiExecuteBuiltIn` function to launch dialogs or perform other operations that are "built-in" to the GUI. Built-in operations are stored for each GUI property, and can be viewed for any particular object using the `guiGetPropertyValues` function. For example, suppose we wanted to view the "About S-PLUS" dialog at some point in our function. Open the Object Browser and create a new page containing the Interface Class Menu Item. Expand the `SPlusMenuBar` node and highlight the menu of interest in the left pane. Right-click on the desired menu item in the right pane and select Command from the right-click menu. The built-in operation is shown at the top of the page:

```
> guiExecuteBuiltIn("$$SPlusMenuBar$Object_Browser$Window$
  Title-Vertical")
```

We can then use this command in a call to `guiExecuteBuiltIn`:

```
> guiExecuteBuiltIn(
  "$$SPlusMenuBar$Object_Browser$Help$About_S_PLUS")
```

## guiGetPropertyOptions

Use the `guiGetPropertyOptions` function to see a list of acceptable values for a given GUI property. For example, you can determine the available border styles for objects of GUI class "Box" as follows:

```
> guiGetPropertyOptions("Box", "BorderStyle")
[1] "None"      "Solid"      "Dots"        "Dot Dash"
[5] "Short Dash" "Long Dash"  "Dot Dot Dash" "Alt Dash"
[9] "Med Dash"  "Tiny Dash"
```

## guiGetPropertyPrompt

Use the `guiGetPropertyPrompt` to see basic information about the property, such as its GUI prompt, its default value, and whether it is a

required property. For example, for the GUI class "Box", the Border Style property information is as follows:

```
> gui GetPropertyPrompt("Box", "BorderStyle")
$PropName:
[1] "BorderStyle"

$prompt:
[1] "Style"

$default:
[1] "Solid"

$optional:
[1] T

$data.mode:
[1] "character"
```

## Identifying Specific Graphics Objects

To modify specific pieces of editable graphics using `gui Modify`, you must specify the object name, showing its path in the object hierarchy. You can use the following functions to get the object name for a specific object type. Most of them take a `GraphSheet` name and a `GraphNum` argument; you can use `gui GetGSName` to obtain the name of the current `GraphSheet`:

- `gui GetAxisLabelName`: returns the name of the `AxisLabels` for a specified axis (axis 1 by default).
- `gui GetAxisName`: returns the name of the axis for a specified axis (axis 1 by default).
- `gui GetAxisTitleName`: returns the name of the axis title for a specified axis (axis 1 by default).
- `gui GetGSName`: returns the name of the current `GraphSheet`. (This function takes no arguments.)
- `gui GetGraphName`: returns the `GraphName` of the graph with the

---

specified GraphNum in the specified GraphSheet.

gui GetPlotName: returns the Name of the plot with the specified PlotNum in the specified GraphNum in the specified GraphSheet.

For example,

```
> gui Plot("Line"DataSetValues=data.frame(1:20, sin(1:20)))
> gui Modify("YAxisTitle", Name=gui GetAxisTitleName(),
  Title="sin(x)")
```

## guiGetPlotClass

Use the gui GetPlotClass function to do one of the following:

1. For a specified plot type, return the GUI class to which the plot type belongs. The class name is a required argument in gui Modify.
2. If no plot type is specified, return a list of valid plot types. These are the valid plot types for gui Plot.

For example,

```
> gui GetPlotClass("Scatter")
[1] "LinePlot"
> gui GetPlotClass()
[1] "Scatter"      "Line"         "LineScatter"
[4] "IsolatedPoints" "HighDensity"  "Text"
[7] "Bubble"       "Color"        "BubbleColor"
[10] "Loess"        "Spline"       "Robust"
[13] "Dot"          "TimeSeries"   "Step"
[16] "VerticalStep" "HorizontalDensity" "YZeroDensity"
[19] "Super"        "Kernel"       "LineNearCF"
[22] "PolynomialCF" "ExpCF"        "LnCF"
...
> gui Plot("Loess", DataSetValues=environmental[, 1:2])
> gui Modify(gui GetPlotClass("Loess"), Name=
  gui GetPlotName(), LineColor="Red")
```

## guiRemoveContents

Use `gui RemoveContents` to remove the objects contained by the specified container.

For example,

```
> gui RemoveContents(" GraphSheet", Name=gui GetGSName)
```

will clear the contents of the current graph sheet, leaving it blank.

## guiUpdatePlots

To update the plots created by `gui Pl ot(DataSetVal ues=...)` with new data set values, use `gui UpdatePl ots`.

For example,

```
> gsName <- gui Pl ot(" Scatter", DataSetVal ues=fuel . frame  
  [, 1: 2])  
> gui UpdatePl ots(GraphSheet=gsName, DataSetVal ues=  
  envi ronmental [, 1: 2])
```

The number of columns in the data set used in `gui UpdatePl ots` should be the same as the number of columns in the original data set used in `gui Pl ot`.

# AUTOMATION IMPROVEMENTS IN S-PLUS 4.5

# 15

---

<b>Passing Data to Functions via Automation</b>	<b>242</b>
Method to get and set parameter classes of functions exposed via automation	243
<b>New Automation Methods in S-PLUS 4.5</b>	<b>246</b>
<b>Automating Embedded S-PLUS Graphs</b>	<b>254</b>
<b>Examples Of Automation Provided With S-plus</b>	<b>255</b>
<b>Examples Of Using S-plus As An Automation Client     Included With S-plus</b>	<b>257</b>
<b>Examples of ActiveX controls included with S-PLUS</b>	<b>258</b>

## PASSING DATA TO FUNCTIONS VIA AUTOMATION

S-PLUS functions that are exposed via automation by calling `register.ole.object()` can be used in other automation client programs to accept data, run the function, and return the resulting data, if any, to the client program.

As discussed earlier, the parameters of a function and the function's return value are properties of the function object and can be used in a client program. You can pass data directly to a function using these properties, and you can retrieve the result of running the function with the `run` method by using the `ReturnValue` property.

For example, if you have a function called `MyFunction(a)` defined in S-PLUS which takes in a data frame and returns a data frame, when you expose this function via automation using `register.ole.object()`, you could use the following Visual Basic 4.0 script to set the function parameter, run the function, and get the return value:

```
Dim pArray(1 to 3) as double
pArray(1) = 1.0
pArray(2) = 2.0
pArray(3) = 3.0

Dim pMyFunction as Object
Set pMyFunction = CreateObject("S-PLUS.MyFunction")
pMyFunction.a = pArray
pMyFunction.Run

Dim pReturnArray as Variant
pReturnArray = pMyFunction.ReturnValue
```

In this example, after the `run` method is called, the function will be executed and the `ReturnValue` property will contain the result of running the function in S-PLUS. This is retrieved in the Visual Basic variable `pReturnArray`.

By default, all parameter data is passed as a data frame to the function. This means that in the above example, `pArray` is first converted into a data frame and then this data frame is passed to the function `MyFunction`. This default behavior could cause errors if the function you've exposed expects data types other than data frames. You can control the data types used in a function exposed via automation in one of two ways. You can call the `SetParameterClasses()` method of the function with a comma-delimited string specifying the data types (or class names) for each of the parameters and the return value of the function.

Alternatively, you can set a property of the `FunctionInfo` object called `ArgumentClassList` with a comma-delimited string specifying the data types (or class names) for each of the parameters and the return value of the function.

**Method to get and set parameter classes of functions exposed via automation:**

**Table 5:**

<b>SetParameterClasses</b>  <pre>[boolean] = obj.SetParameterClasses ( [comma delimited string] )</pre>	Returns TRUE if successful, FALSE if not.	Takes in a comma-delimited string specifying the class names of the return value followed by each of the parameters of the function. It is required that the return value class name be specified first and that the number of class names specified in this string match the total number of parameters of the function plus its return value.
<b>GetParameterClasses</b>  <pre>[string array] = obj.GetParameterClasses ()</pre>	Returns an array of strings representing the class names of the return value followed by each of the parameters of the function.	Takes no parameters. Returns an array of strings representing the class names of the return value followed by each of the parameters of the function.

The above example in Visual Basic 4.0 can be modified to show how to use `SetParameterClasses`. We can use `SetParameterClasses` to adjust how the data from Visual Basic is interpreted by `MyFunction`:

```
Dim pArray(1 to 3) as double
pArray(1) = 1.0
pArray(2) = 2.0
pArray(3) = 3.0

Dim pMyFunction as Object
Set pMyFunction = CreateObject("S-PLUS.MyFunction")
```

```
if ( _
    pMyFunction.SetParameterClasses("data.frame, vector")
= TRUE _
) then
pMyFunction.a = pArray
pMyFunction.Run

Dim pReturnArray as Variant
pReturnArray = pMyFunction.ReturnValue
end if
```

As an alternative to using `SetParameterClasses` in the automation client at the time of running or using the function, you can define the parameter classes using the `ArgumentClassList` property when you define the `FunctionInfo` object to represent the function in S-PLUS. This approach has the advantage of simplifying the automation client program code but does require some additional steps in S-PLUS when defining the function.

Consider the following S-PLUS script to define the function `MyFunction` and a `FunctionInfo` object for this function:

```
MyFunction <- function(a)
{
    return(a)
}

guiCreate(
    "FunctionInfo", Function = "MyFunction",
    ArgumentClassList = "vector, vector" );
```

This example script will define `MyFunction` and will define a `FunctionInfo` object for `MyFunction` and set the `ArgumentClassList` to the string "vector, vector" indicating that data passed into and out of `MyFunction` via automation will be done using S-PLUS vectors.

If this is done, then the corresponding Visual Basic 4.0 code becomes simpler because we no longer need to set the parameter classes for the function before it is used:

```
Dim pArray(1 to 3) as double
pArray(1) = 1.0
pArray(2) = 2.0
pArray(3) = 3.0

Dim pMyFunction as Object
Set pMyFunction = CreateObject("S-PLUS.MyFunction")
```



```
pMyFunction.a = pArray  
pMyFunction.Run
```

```
Dim pReturnArray as Variant  
pReturnArray = pMyFunction.ReturnValue
```

pArray is now passed as a **vector** to MyFunction and pReturnArray is now retrieved from MyFunction as a **vector**.

# NEW AUTOMATION METHODS IN S-PLUS 4.5

There are several new automation methods in S-PLUS 4.5:

**Table 6:**

**Object Dialog Methods**

**ShowDialogInParent**

```
[boolean] =  
obj . ShowDialogInParent(  
  [hwnd] )
```

Displays a modal property dialog for an object in the automation client interface. The client program is paused while the dialog is displayed.

Takes in a long number representing the window handle of the window you want the object dialog to appear inside. Returns TRUE if successful and FALSE if not.

**ShowDialogInParentModeless**

```
[boolean] =  
obj . ShowDialogInParentModeless(  
  [hwnd] )
```

Displays a modeless property dialog for an object in the automation client interface. The client program continues executing while the dialog is displayed.

Takes in a long number representing the window handle of the window you want the object dialog to appear inside. Returns TRUE if successful and FALSE if not.

**Object Methods**

**ObjectContainees**

```
[array objects] =  
obj . ObjectContainees(  
  [class name string] )
```

Returns an array of objects that are contained by this object.

This method is not available for function objects.

Takes in a string representing the class name of objects to include in this array. Returns an array of containee objects of the class name specified or an empty array if none can be found.

**Table 6:**

<b>ObjectContainer</b>  <pre>[object] = obj.ObjectContainer()</pre>	Returns an object that is the container of this object.  This method is not available for function objects.	Takes no parameters. Returns the object that contains this object.
<b>ClassName</b>  <pre>[string] = obj.ClassName()</pre>	Returns a string representing the class name of this object.	Takes no parameters. Returns the object class name as a string.
<b>PathName</b>  <pre>[string] = obj.PathName()</pre>	Returns a string representing the path name of this object in S-PLUS.	Takes no parameters. Returns the object path name in S-PLUS.
<b>Application Object Methods</b>		
<b>ExecuteStringResult</b>  <pre>[string] = obj.ExecuteStringResult(     [S-PLUS syntax     string],     [boolean] )</pre>	Returns a string representing the output from executing the string passed in. The format of the return string depends on the setting of the second parameter. If TRUE, the older S-PLUS 3.3 output formatting will be applied. If FALSE, the new format will be used.	Takes in a string representing any valid S-PLUS syntax and a boolean parameter indicating how the result should be formatted. Returns a string representing the result of executing the syntax passed in.
<b>SetSAPIObject</b>  <pre>[boolean] = obj.SetSAPIObject(     [byte array variant],     [object name string] )</pre>	Sets a binary SAPI object created in an automation client program into S-PLUS, making it available to other operations in S-PLUS.	Takes in a VARIANT representing a byte array of the SAPI object to set, and a string representing the name of the object to set. Returns TRUE if successful, otherwise FALSE.

**Table 6:**

<p>GetSAPIObject</p> <pre>[byte array variant] = obj.GetSAPIObject(     [object name string] )</pre>	<p>Returns a binary SAPI object into a variant byte array given the name of the object in S-PLUS.</p>	<p>Takes in a string representing the name of the SAPI object to return. Returns a variant byte array representing the object found. If no object could be found the variant will be empty.</p>
<p>Function Object Methods</p>		
<p>SetParameterClasses</p> <pre>[boolean] = obj.SetParameterClasses(     [comma delimited string] )</pre>	<p>Returns TRUE if successful, FALSE if not.</p>	<p>Takes in a comma-delimited string specifying class names of the return value followed by each parameter of the function. It is required that the return value class name be specified first and that the number of class names specified in this string match the total number of parameters of the function plus its return value.</p>
<p>GetParameterClasses</p> <pre>[string array] = obj.GetParameterClasses() )</pre>	<p>Returns an array of strings representing the class names of the return value followed by each of the parameters of the function.</p>	<p>Takes no parameters. Returns an array of strings representing the class names of the return value followed by each of the parameters of the function.</p>
<p>GraphSheet Object Methods</p>		

Table 6:

<p>CreatePlots</p> <pre>[boolean] = obj.CreatePlots(   [axis type string],   [plot type string],   [data array variant],   [data column names   array] )</pre>	<p>Returns TRUE if successful, FALSE if not.</p>	<p>Takes in a string representing the axis type for the plots, a string representing the plot type to create, and the data to use to create plots in the graphsheet.</p> <p>The axis type string may be one of “2D”, “3D”, “Pie”, or “Polar”.</p> <p>The plot type string is one of the choices shown in the Insert Graph dialog (accessed by the menu item Insert/Graph in S-PLUS).</p> <p>The last parameter is an array of strings representing the names of columns in the data array. These names will be used as axes labels in plots. Pass in an empty variant to not use column names.</p>
<p>CreateConditionedPlots</p> <pre>[boolean] = obj.CreateConditionedPlots(   [axis type string],   [plot type string],   [number of   conditioning vars],   [data array variant],   [data column names   array] )</pre>	<p>Returns TRUE if successful, FALSE if not.</p>	<p>Similar to CreatePlots except that this method takes in a number specifying the number of conditioning columns to use from the data array passed in. The data columns and conditioning columns are specified as part of the data array passed in.</p>

**Table 6:**

CreateConditioned- PlotsSeparateData	Returns TRUE if suc- cessful, FALSE if not.	Similar to Create- Conditioned- Plots except that this method takes in a data array and a conditioning array separately, instead of combined in one ar- ray.
<pre>[boolean] = obj. CreateConditionedPlotsS eparateData(   [axis type string],   [plot type string],   [data array variant],   [conditioning array variant],   [data column names array],   [conditioning col. names array])</pre>		The last two parameters are arrays of strings rep- resenting the names of columns in the data ar- ray and names of col- umns in the conditioning array. These names will be used as axes labels in plots. Pass in an empty variant for either or both of these to not use col- umn names.

Table 6:

<div>CreatePlotsGallery</div> <div><pre>[boolean] = obj.CreatePlotsGallery(   [hwnd],   [data array variant],   [data column names array] )</pre></div>	<div>Returns TRUE if suc- cessful, FALSE if not.</div>	<div>Displays a dialog allow- ing selection of axis type and plot type. Takes in a long number representing the win- dow handle of the win- dow you want the graph gallery dialog to appear inside, and a data array to plot. Returns TRUE if successful and FALSE if not.</div> <div>The last parameter is an array of strings repre- senting the names of columns in the data ar- ray. These names will be used as axes labels in plots. Pass in an empty variant to not use col- umn names.</div>
<div>CreateConditionedPlots- Gallery</div> <div><pre>[boolean] = obj.CreateCondi ti onedPl otsGallery(   [number of condi ti oning vars],   [hwnd],   [data array variant],   [data column names array] )</pre></div>	<div>Returns TRUE if suc- cessful, FALSE if not.</div>	<div>Similar to Cre- atePlotsGallery except that this method takes in a number speci- fying the number of conditioning columns to use from the data array passed in. The data col- umns and conditioning columns are specified as part of the data array passed in.</div>

**Table 6:**

CreateConditionedPlots – SeparateDataGallery	Returns TRUE if suc- cessful, FALSE if not.	Similar to Create- Conditioned- PlotsGallery except that this method takes in a data array and a conditioning array separately, instead of combined in one array.
<pre>[boolean] = obj.CreateCondi tionedPl ots -     SeparateDataGal lery(         [hwnd],         [data array variant],         [condi tioning array variant],         [data column names array],         [condi tioning col. names array])</pre>		The last two parameters are arrays of strings rep- resenting the names of columns in the data ar- ray and names of col- umns in the conditioning array. These names will be used as axes labels in plots. Pass in an empty variant for either or both of these to not use col- umn names.

Examples of using these new automation methods can be found in the following sub-directories under **samples/oleauto** in your S-PLUS program directory:

**dialogs**

This directory contains an example Visual Basic 4.0 project that demonstrates the use of ShowDialog(), ShowDialogInParent() and ShowDialogInParentModeless() automation methods.

**objects**

This directory contains an example Visual Basic 4.0 project that demonstrates the use of ObjectContainees(), ObjectContainer(), ClassName(), and PathName() automation methods.

**createplt**



This directory contains an example Visual Basic 4.0 project that demonstrates the use of the `CreatePlots()`, `CreateConditionedPlots()`, and `CreateConditionedPlotsSeparateData()` automation methods.

## AUTOMATING EMBEDDED S-PLUS GRAPHS

With S-PLUS Automation support you can automate embedded graph sheet documents easily in any Automation client program such as Visual Basic, Excel, Word, and others. You can even create, modify, and save an embedded S-PLUS graph sheet with plotted data in place without ever leaving your Automation client program. You can use S-PLUS Automation to create a plot, send data from your Automation client to the plot, modify properties of the plot either through dialogs which you can display in your client program or directly via command, execute S-PLUS built-in functions or functions that you've written in the S-PLUS language and exposed via Automation to transform the data, and save the plot with your Automation client document. Examples written in Visual Basic 4.0 and in Visual Basic for Applications with Excel 7.0 are distributed with S-PLUS to demonstrate automating an embedded graph sheet.

'VBEMBED.EXE' and corresponding Visual Basic source files can be found in 'samples/oleauto/vbembed' off your S-PLUS program directory. This example demonstrates how to do the following:

- embed an S-PLUS graph sheet
- add objects to it
- modify those objects by displaying object property dialogs in the client program,
- delete objects from it,
- save a document containing the embedded graph sheet. 'P

'PLOTDATA.XLS' can be found in 'samples/oleauto/vba'. This example demonstrates how to do the following:

- embed an S-PLUS graph sheet
- add a plot to it
- send Excel data from a worksheet to S-PLUS to be graphed in the plot
- modify plot properties using property dialogs.

# EXAMPLES OF AUTOMATION PROVIDED WITH S-PLUS

The following directories (all paths relative to the S-PLUS installation directory) contain examples of using S-PLUS as an automation server included with S-PLUS:

## **samples/oleauto**

**senddata** Example Visual Basic 4.0 project that shows how to send data to S-PLUS data objects.

**vbembed** Example Visual Basic 4.0 project that shows how to embed an S-PLUS graphsheet, modify it by using automation, save it, delete objects in it, and how to display an object dialog.

**vbclient** Example Visual Basic 4.0 project that demonstrates creating a graphsheet, adding an arrow to it, changing the properties of the arrow, showing a dialog for the arrow, executing S-PLUS commands, modifying option values, getting an object, and sending and receiving data.

**vba** This directory contains several examples of using Visual Basic for Applications in Excel:

**auto\_vba.xls:** Demonstrates sending and receiving data, and converting Excel ranges to arrays.

**plotdata.xls:** Demonstrates embedding a graphsheet and adding and modifying a plot in it.

**xfertodf.xls:** Demonstrates transferring Excel ranges to S-PLUS dataframes and back to Excel.

**vbrunfns** This directory contains an example Visual Basic 4.0 project that shows how to register an S-PLUS function as automatable, how to pass binary data to the function, and how to receive the result of the function back in VB.

**dialogs** This directory contains an example Visual Basic 4.0 project that demonstrates the use of the following automation methods:

ShowDialog()  
ShowDialogInParent()

	ShowDialog(nParentModel ess())
<b>objects</b>	This directory contains an example Visual Basic 4.0 project that demonstrates the use of ObjectContainees(), ObjectContainer(), ClassName(), and PathName() automation methods.
<b>creatept</b>	This directory contains an example Visual Basic 4.0 project that demonstrates the use of the CreatePlots() automation method.

---

## EXAMPLES OF USING S-PLUS AS AN AUTOMATION CLIENT INCLUDED WITH S-PLUS

The following directories (all paths relative to the S-PLUS installation directory) contain examples of using S-PLUS as an automation client:

### **samples/oleauto**

<b>clitest.a.ssc</b>	Shows how to use S-PLUS commands to start Excel and call method of Excel to convert inches to points and return the result in S-PLUS.
<b>clitest.b.ssc</b>	Shows how to use S-PLUS commands to start Excel, get a property, and set a property.
<b>clitest.c.ssc</b>	Shows how to use S-PLUS commands to set a range of data in an Excel worksheet with data from an S-PLUS vector and then how to get the data back from Excel into another vector.
<b>clitest.d.ssc</b>	Shows how to get a property value from Excel
<b>clitest.e.ssc</b>	Shows how to send a vector from S-PLUS to Excel and transpose it to a row in Excel.
<b>clitest.f.ssc</b>	Shows how to send a vector from S-PLUS to Excel and transpose it to a row in Excel using a different set of steps than in <b>clitest.e.ssc</b> .

## EXAMPLES OF ACTIVEX CONTROLS INCLUDED WITH S-PLUS

Examples of ActiveX controls which implement support for S-PLUS dialog containment are provided on disk in the **SAMPLES/OCX** directory beneath the program directory. These examples are C++ projects in Microsoft Visual C++ 4.1 using MFC (Microsoft Foundation Classes) and are intended for developers. `samples/ocx`

<b>myocx</b>	Microsoft Visual C++ 4.1 MFC project demonstrating how to write ActiveX controls that fully support S-PLUS dialogs.
<b>ocx1</b>	Microsoft Visual C++ 4.1 MFC project demonstrating how to write ActiveX controls that fully support S-PLUS dialogs.
<b>support</b>	Microsoft Visual C++ 4.1 MFC headers and source files necessary for making ActiveX controls that fully support S-PLUS dialogs.

# DIALOG CONTROLS IN S-PLUS 4.5

# 16

---

<b>ActiveX Controls in S-PLUS dialogs</b>	<b>260</b>
Adding an ActiveX control to a dialog	260
Where can the PROGID for the control be found?	261
Registering an ActiveX control	263
Why only “OCX String”?	264
Common error conditions when using ActiveX controls in S-PLUS	264
Designing ActiveX controls that support S-PLUS	265
<b>New Dialog Controls In S-PLUS 4.5</b>	<b>279</b>

## ACTIVE X CONTROLS IN S-PLUS DIALOGS

S-PLUS supports the use of ActiveX controls in dialogs for user defined functions created in the S-PLUS programming language. This feature allows greater flexibility when designing a dialog to represent a function and its parameters. Any ActiveX control can be added to the property list for a dialog, however, most ActiveX controls will not automatically communicate changed data back to the S-PLUS dialog nor will most tell S-PLUS how much space to give the control in the dialog. To fully support S-PLUS dialog layout and data communication to and from S-PLUS dialogs, a few special ActiveX methods, properties, and events need to be implemented in the control by the control designer.

Examples of ActiveX controls which implement support for S-PLUS dialog containment are provided on disk in the SAMPLES/OCX directory beneath the program directory. These examples are C++ projects in Microsoft Visual C++ 4.1 using MFC (Microsoft Foundation Classes). Any MFC ActiveX project can be modified to support S-PLUS dialogs easily, and this will be discussed later in this section. Also in SAMPLES/OCX are example scripts which use S-PLUS to test these ActiveX controls.

### Adding an ActiveX control to a dialog

To use an ActiveX control for a property in a dialog, when creating the property, specify a "DialogControl" of type "OCX String" and specify the program id (or PROGID) of the control using the "ControlProgId" subcommand. Below is an example S-PLUS script which creates a property that uses an ActiveX control:

```
gui Create("Property",
  name = "OCXStringField",
  DialogControl = "OCX String",
  ControlProgId = "TXTESTCONTROL1.TxTestControl1Ctrl.1",
  ControlServerPathName = "c:\\myocx\\myocx.ocx",
  DialogPrompt = "&OCX String");
```

If you are editing or creating a property using the object browser, the Property object dialog for the property you are editing allows you to set the dialog control type to "OCX String" from the "Dialog Control" drop-down list. When this is done, the "Control ProgId" and "ControlServerPathName" fields become enabled allowing you to enter the PROGID of the ActiveX control and its location on disk, respectively. The "ControlServerPathName" value is used to autoregister the control, if necessary, before using the control.

If you are editing or creating a property using the object browser, the



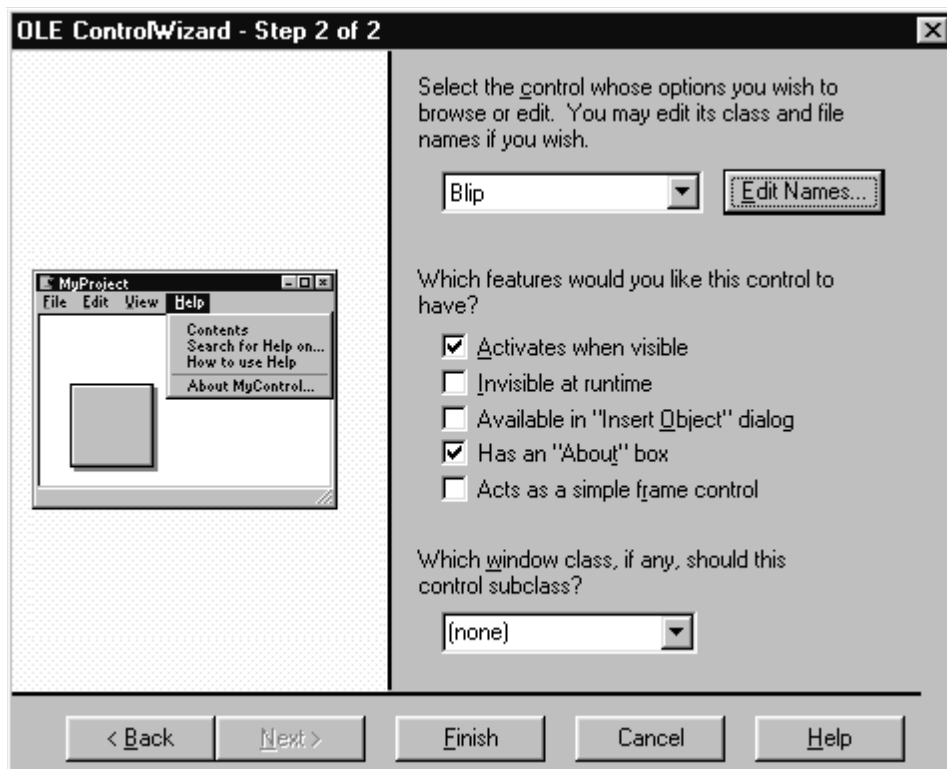
Property object dialog for the property you are editing allows you to set the dialog control type to “OCX String” from the “Dialog Control” drop-down list. When this is done, the “Control ProgId” field becomes enabled allowing to you enter the PROGID of the ActiveX control.

**Where can the  
PROGID for the  
control be  
found?**

When you add an ActiveX control to an S-PLUS dialog, you need to specify its PROGID, as mentioned above. The PROGID is a string which uniquely identifies this control on your system. If you create controls using the ControlWizard in Developer Studio as part of Microsoft Visual C++ 4.0 or higher, a default value for the PROGID is created by the ControlWizard during control creation that is based on the name of the project you use. For example, if your ControlWizard project name is “MyOCX”, then the PROGID that is generated is “MYOCX.MyOCXCtrl.1”. The pattern here is [Project name].[Control class name without the leading ‘C’].1. You can also find the PROGID used in an MFC ControlWizard project in the implementation CPP file of the control class. Search for the `IMPLEMENT_OLECREATE_EX()` macro in this file. The second parameter in this macro is the PROGID string you are looking for.

If you are using the OLE ControlWizard as part of Microsoft Visual C++ 4.0 or higher to develop your control, you can change the PROGID string for your control before it gets created by editing the names used for the control project. During the ControlWizard steps, you will see a dialog with the

button “Edit Names” on it:



Click on this button and you will get another dialog allowing you to change the names used for classes in this project. Every control project in MFC has a class for the control and a class for the property sheet for the control. In the control class section of this dialog you will see the “Type ID” field. This is

the PROGID for the control:

**Edit Names**

Short Name:

Control

Class Name:	Header File:	Type Name:
<input type="text" value="CBlipCtrl"/>	<input type="text" value="BlipCtrl.h"/>	<input type="text" value="BlipCtrl"/>
Implementation File:	Type ID:	
<input type="text" value="BlipCtrl.cpp"/>	<input type="text" value="BLIP.BlipCtrl.1"/>	

Property Page

Class Name:	Header File:	Type Name:
<input type="text" value="CBlipPropPage"/>	<input type="text" value="BlipPpg.h"/>	<input type="text" value="Blip Property Page"/>
Implementation File:	Type ID:	
<input type="text" value="BlipPpg.cpp"/>	<input type="text" value="BLIP.BlipPropPage.1"/>	

OK  
Cancel  
Help

## Registering an ActiveX control

It is important to register an ActiveX control with the operating system at least once before using it so that whenever the PROGID of the control is referred to (such as in the “ControlProgId” subcommand above), the operating system can properly locate the control on your system and run it. Registering an ActiveX control is usually done automatically during the creation of the control, such as in Microsoft Visual C++ 4.0 or higher. If the subcommand “ControlServerPathName” is specified in an S-PLUS script using the control, then this value will be used to register the control automatically. A control can also be registered manually by using a utility called “RegSvr32.exe”. This utility is included with development systems that support creating ActiveX controls, such as Microsoft Visual C++ 4.0 or higher. For your convenience, a copy of RegSvr32.exe is located in the SAMPLES/OCX directory, along with two useful batch files, “RegOCX.BAT” and “UnRegOCX.BAT”, which will register and unregister a control. You can modify these batch files for use with controls you design.

You typically do not ever need to unregister an ActiveX control, unless you wish to remove the control permanently from your system and no longer need to use it with any other container programs such as S-PLUS. If this is the case, you can use RegSvr32.exe with the '/u' command line switch (as in UnRegOCX.BAT) to unregister the control.

### **Why only “OCX String”?**

In S-PLUS, several different types of properties exist. There are string, single-select lists, multi-select lists, numeric, and others. This means that a property in a dialog communicates data depending on the type of property selected. A string property communicates string data to and from the dialog. A single-select list property communicates a number representing the selection from the list, a multi-select list communicates a string of selections made from the list with delimiters separating the selections. For ActiveX controls, only string communication has been provided in this version. This means that the control should pass a string representing the “value” or state of the control back to S-PLUS. In turn, if S-PLUS needs to change the state of the control, it will communicate a string back to the control. Using a string permits the most general type of communication between S-PLUS and the ActiveX control, because so many different types of data can be represented with a string, even for example lists. In future versions, other S-PLUS property types may be added for ActiveX controls.

### **Common error conditions when using ActiveX controls in S-PLUS**

The most common problem when using an ActiveX control in an S-PLUS dialog is that the control does not appear, instead a string edit field shows up when the dialog is created. This is usually caused by not registering the ActiveX control with the operating system. After a control is first created and before it is ever used, it must be registered with the operating system. This usually occurs automatically in the development system used to make the control, such as Microsoft Visual C++. However, you can also manually register the control by using a utility called “RegSvr32.exe”. This utility is included with development systems that support creating ActiveX controls, such as Microsoft Visual C++ 4.0 or higher. For your convenience, a copy of RegSvr32.exe is located in the SAMPLES/OCX directory, along with two useful batch files “RegOCX.BAT” and “UnRegOCX.BAT” which will register and unregister controls. You can modify these batch files for use with controls you design.

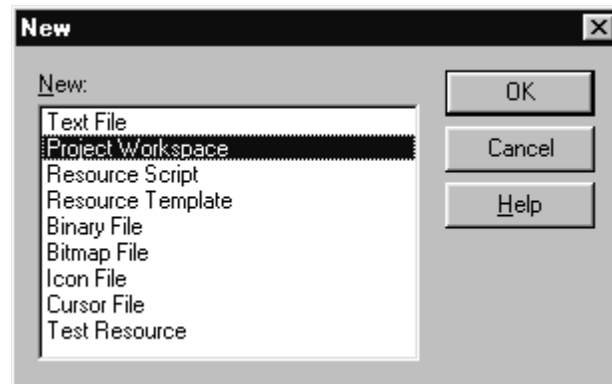
## Designing ActiveX controls that support S-PLUS

As mentioned earlier, examples of ActiveX controls which implement support for S-PLUS are provided on disk in the SAMPLES/OCX directory beneath the program directory. One of the examples in this directory is called MyOCX, and it is a C++ project in Microsoft Visual C++ 4.1 using MFC. There is also an example S-PLUS script in MyOCX which shows how to use this ActiveX control in an S-PLUS dialog. This example will be used here to show how to implement ActiveX controls for S-PLUS. If you would rather skip this section and simply study the changes in the source files for MyOCX, all changes are marked in the source files with the step number (as listed below) that the change corresponds to. Just search for the string “S-PLUS Dialog change (STEP” in all the files of the MyOCX project to find these modifications.

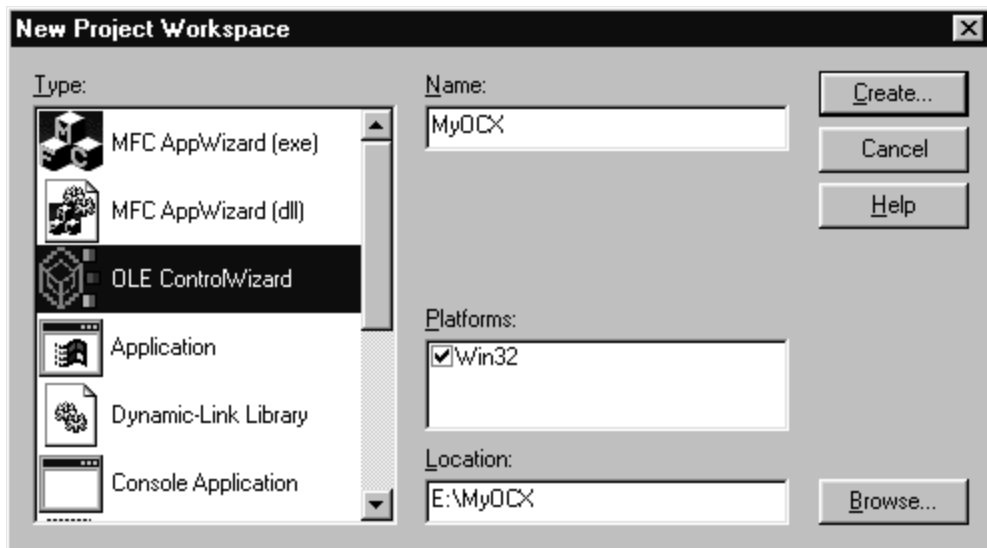
Version 4.0 or higher of Microsoft Visual C++ is used to demonstrate ActiveX control creation. Higher versions can also be used to create controls for S-PLUS but the dialogs and screens shown may be different.

### 1. Create the basic control

The first step to designing an ActiveX control in MFC should be to use the OLE ControlWizard that is part of the Developer Studio. Select New from the File menu in Developer Studio and then choose “Project Workspace” to start a new project.

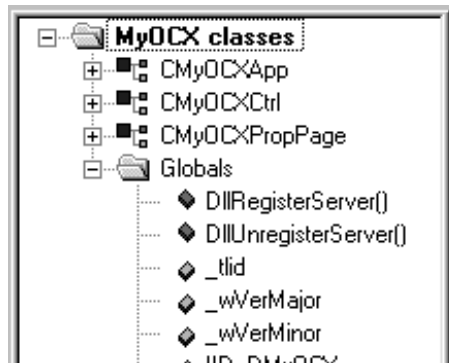


From the workspace dialog that appears, select “OLE ControlWizard” from the list of workspace types available. Enter a name for the project and specify the location, then click the “Create...” button.



After accepting this dialog, you will see a series of dialogs associated with the OLE ControlWizard, asking questions about how you want to implement your control. For now, you can simply accept the defaults by clicking “Next” on each dialog. When you reach the last dialog, click the “Finish” button. You will see a confirmation dialog showing you the choices you selected and names of classes that are about to be created. Click the “OK” button to accept and generate the project files.

In the “ClassView” page of the “Project Workspace” window in Visual C++, you will see the classes that the OLE ControlWizard created for your ActiveX control:



## 2. Add the S-PLUS support classes

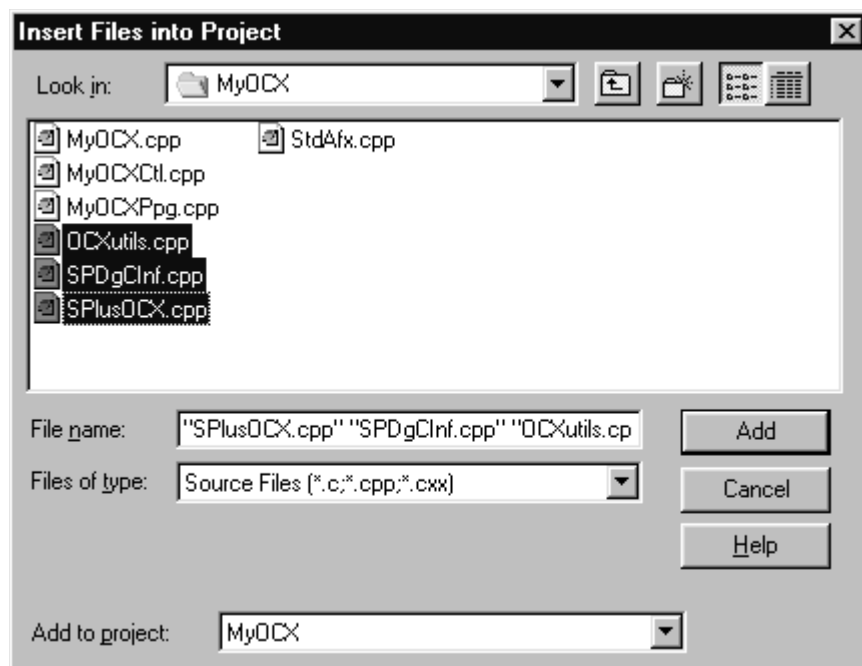
To start adding support for S-PLUS dialogs to your ActiveX control, copy the following files from the SAMPLES/OCX/SUPPORT control example directory into the new ActiveX control project directory you just created:

OCXUtils.cpp  
 OCXUtils.h  
 SPDgCInf.cpp  
 SPDgCInf.h  
 SPlusOCX.cpp  
 SPlusOCX.h  
 SPlusOCX.idl

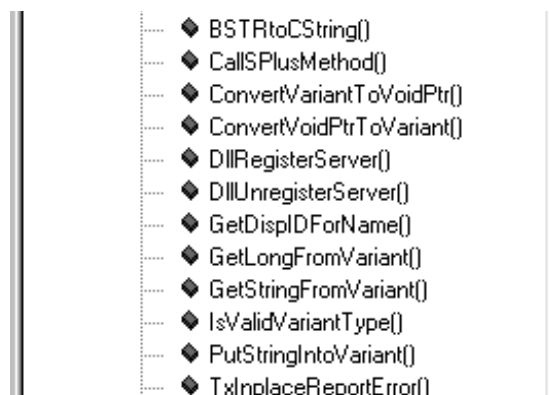
You also need to add these classes to your project before they will be compiled and linked to your control. To do this, select “Files into Project...” from the “Insert” menu in Visual C++. You will then see a standard file open dialog. Use this dialog to select the following files:

OCXUtils.cpp  
 SPDgCInf.cpp  
 SPlusOCX.cpp

To select all these files at once, hold down the CTRL key while using the mouse to click on the filenames in the list.



When these files are selected, click the “Add” button and the classes will appear as entries in your Project Workspace window.

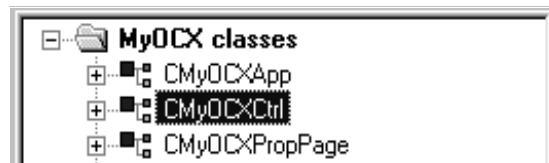




### 3. Modify class inheritance

Next, we need to modify the inheritance of the class representing your ActiveX control so that it inherits from CSPlusOCX instead of from COleControl. CSPlusOCX is a parent class from which all ActiveX controls for which you desire support for S-PLUS dialogs can inherit. CSPlusOCX inherits directly from COleControl and its complete source code can be found in the SPlusOCX.cpp and SPlusOCX.h files.

To do this, first double-click on the class representing your ActiveX control in the “ClassView” page of the Project Workspace window to open the header for this class into your editor. In this example that is the CMyOCXCtrl class. Go to the top of this file in the editor.



Add the following line before the class declaration line for CMyOCXCtrl at the top of this header file:

```
#include "SPlusOCX.h"
```

Modify the class declaration line

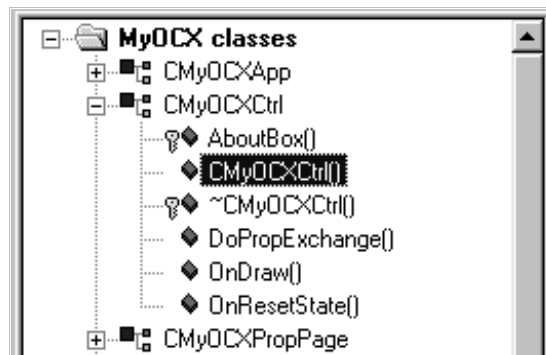
```
class CMyOCXCtrl : public COleControl
```

to read

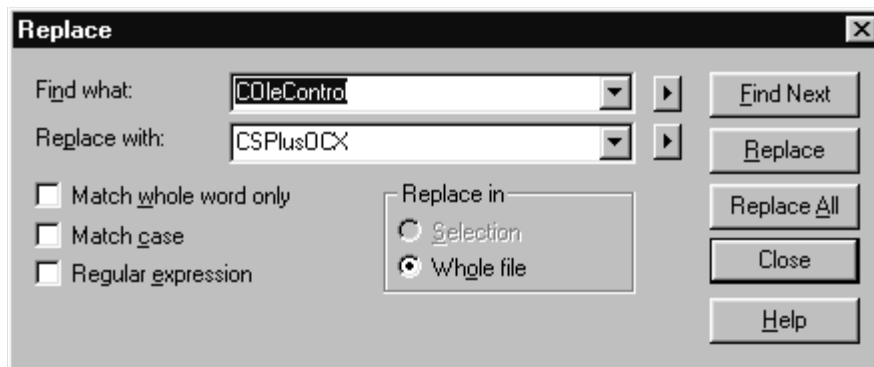
```
class CMyOCXCtrl : public CSPlusOCX
```

Next, expand the class listing for CMyOCXCtrl so that all the methods are shown. To do this, click on the ‘+’ next to “CMyOCXCtrl” in the “ClassView” page of the

Project Workspace window.

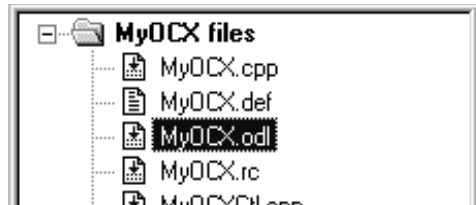


Then double-click on the constructor “CMyOCXCtrl ( )” to open the implementation CPP file for this class in your editor. Go to the top of this file. Using the find and replace function of the Developer Studio, replace all occurrences of “COleControl” base class with the new base class name “CSPPlusOCX” in this file:



#### 4. Modify your control's type library definition file

Switch to the “FileView” page in the Project Workspace window and find the type library definition file (.ODL) for your ActiveX control. In this example it is “My-OCX.odl”. Double-click on this entry in the list to open this file into your editor. Go to the top of this file.



Find the “properties” definition section for the dispatch interface “\_DMyOCX” in this file. It should look like:

```
dispinterface _DMyOCX
{
    properties:
        // NOTE - ClassWizard will maintain property information here.
        // Use extreme caution when editing this section.
        //{{AFX_ODL_PROP(CMyOCXCtrl)
        //}}AFX_ODL_PROP
```

Add the following lines at the end of this section:

```
#define SPLUSOCX_PROPERTIES
#include "SPlusOCX.idl"
#undef SPLUSOCX_PROPERTIES
```

The section should now appear as follows:

```
dispinterface _DMyOCX
{
    properties:
        // NOTE - ClassWizard will maintain property information here.
        // Use extreme caution when editing this section.
        //{{AFX_ODL_PROP(CMyOCXCtrl)
        //}}AFX_ODL_PROP

        #define SPLUSOCX_PROPERTIES
        #include "SPlusOCX.idl"
        #undef SPLUSOCX_PROPERTIES
```

```
methods:
// NOTE - ClassWizard will maintain method information here.
//   Use extreme caution when editing this section.
//{{AFX_ODL_METHOD(CMyOCXCtrl)
//}}AFX_ODL_METHOD
```

```
[id(DISPID_ABOUTBOX)] void AboutBox();
};
```

Now, add the following lines at the end of the “methods” section just below the “properties” section you just modified:

```
#define SPLUSOCX_METHODS
#include "SPlusOCX.idl"
#undef SPLUSOCX_METHODS
```

This whole section should now appear as follows:

```
dispinterface _DMyOCX
{
properties:
// NOTE - ClassWizard will maintain property information here.
//   Use extreme caution when editing this section.
//{{AFX_ODL_PROP(CMyOCXCtrl)
//}}AFX_ODL_PROP
```

```
#define SPLUSOCX_PROPERTIES
#include "SPlusOCX.idl"
#undef SPLUSOCX_PROPERTIES
```

```
methods:
// NOTE - ClassWizard will maintain method information here.
//   Use extreme caution when editing this section.
//{{AFX_ODL_METHOD(CMyOCXCtrl)
//}}AFX_ODL_METHOD
```

```
[id(DISPID_ABOUTBOX)] void AboutBox();
```

```
#define SPLUSOCX_METHODS
#include "SPlusOCX.idl"
#undef SPLUSOCX_METHODS
```

```
};
```

Next, locate the event dispatch interface sections. In this example, it appears as:

```
dispinterface _DMyOCXEvents
{
properties:
// Event interface has no properties

methods:
// NOTE - ClassWizard will maintain event information here.
// Use extreme caution when editing this section.
//{{AFX_ODL_EVENT(CMyOCXCtrl)
//}}AFX_ODL_EVENT
};
```

Add the following lines in the “events” section:

```
#define SPLUSOCX_EVENTS
#include "SPlusOCX.idl"
#undef SPLUSOCX_EVENTS
```

The section should now appear as:

```
dispinterface _DMyOCXEvents
{
properties:
// Event interface has no properties

methods:
// NOTE - ClassWizard will maintain event information here.
// Use extreme caution when editing this section.
//{{AFX_ODL_EVENT(CMyOCXCtrl)
//}}AFX_ODL_EVENT

#define SPLUSOCX_EVENTS
#include "SPlusOCX.idl"
#undef SPLUSOCX_EVENTS

};
```

Do not modify any other parts of this file at this time.

## 5. Build the control

Now is a good time to build this project. To do this, click on the “Build” toolbar button or select “Build MyOCX.OCX” from the “Build” menu in the Developer Studio. If you receive any errors, go back through the above steps to make sure you have completed them correctly. You may receive warnings:

```
OCXutils.cpp(125) : warning C4237: nonstandard extension used : 'bool' keyword
is reserved for future use
```

```
OCXutils.cpp(216) : warning C4237: nonstandard extension used : 'bool' keyword
is reserved for future use
```

These warnings are normal and can be ignored.

Several overrides of CPlusOCX virtual methods still remain to be added to your ActiveX control class, but compiling and linking now gives you a chance to review the changes made and ensure that everything builds properly at this stage.

## 6. Add overrides of virtual methods to your control class

To support S-PLUS dialog layout and setting the initial value of the control from an S-PLUS property value, you need to override and implement several methods in your control class. To do this, edit the header for your control class. In this example, edit the “MyOCXCtrl.h” file. In the declaration of the CMyOCXCtrl class, add the following method declarations in the “public” section:

```
virtual long GetSPlusDialogVerticalSize( void );
virtual long GetSPlusDialogHorizontalSize( void );
virtual BOOL SPlusOnInitializeControl(const VARIANT FAR&
    vInitialValue);
```

Next, open the implementation file for your control class. In this example, edit the file “MyOCXCtrl.cpp”. Add the following methods to the class:

```
long CMyOCXCtrl::GetSPlusDialogVerticalSize()
{
    return 3; // takes up 3 lines in dialog
}
```

```
long CMyOCXCtrl::GetSPlusDialogHorizontalSize()
{
    return 1; // takes up 1 column in dialog
}
```

```
BOOL CMyOCXCtrl::SPlusOnInitializeControl(const VARIANT
    FAR& vInitialValue)
```

```

{
    CString sInitialValue; sInitialValue.Empty();
    if ( GetStringFromVariant(
        sInitialValue,
        vInitialValue,
        "InitialValue" ) )
    {
        // Set properties here
    }

    return TRUE;
}

```

These three methods should be implemented in the control class of any ActiveX control supporting S-PLUS dialogs fully. The first two methods support dialog layout, while the third supports setting values for the control from S-PLUS.

The value returned by `GetSPlusDialogVerticalSize()` should be a long number representing the number of lines the control takes up in an S-PLUS dialog. A line is the size of an String edit field property in an S-PLUS dialog. The value returned by `GetSPlusDialogHorizontalSize()` should be either 1 or 2. Returning 1 means that this control takes up only one column in an S-PLUS dialog. Returning 2 means the control takes up two columns. A column in an S-PLUS dialog is the width of a single String property field. There are at most two columns in an S-PLUS dialog. In the example above, the `MyOCX` control takes up three lines and only one column in an S-PLUS dialog.

`SPlusOnInitializeControl()` is called when the control is first enabled in the S-PLUS dialog and every time the property that this control corresponds to in S-PLUS is changed. It receives a variant representing the initial value or current value (if any) for the control. This method should return `TRUE` to indicate successful completion and `FALSE` to indicate failure. Included in the file “`OCX-Utils.h`” (copied previously into your control project directory) are numerous helper functions such as the one used here `GetStringFromVariant()` which will convert the incoming variant into a string if possible. You can then use this string to set one or more properties in your control.

To use the `SPlusOnInitializeControl()` in this example ActiveX control, first add a member string to the control class. Edit the “`MyOCXCtrl.h`” file and add a `CString` member variable called “`m_sValue`” to the `CMyOCXCtrl` class:

```

private:
    CString m_sValue;

```

Next, initialize this value in the constructor for `CMyOCXCtrl` by modifying the

constructor definition in “MyOCXCtrl.cpp”:

```
CMyOCXCtrl::CMyOCXCtrl()
{
    InitializeIIDs(&IID_DMyOCX, &IID_DMyOCXEvents);

    // TODO: Initialize your control's instance data here.

    m_sValue.Empty();
}
```

Then, add lines to the definition of the override of `SPlusOnInitializeControl()` in your control class to set this member variable and refresh the control by modifying “MyOCXCtrl.cpp”:

```
BOOL CMyOCXCtrl::SPlusOnInitializeControl
(const VARIANT FAR& vInitialValue)
{
    CString sInitialValue; sInitialValue.Empty();
    if ( GetStringFromVariant(
        sInitialValue,
        vInitialValue,
        "InitialValue" ) )
    {
        // Set properties here

        m_sValue = sInitialValue;
        Refresh();
    }

    return TRUE;
}
```

Finally, so we can see the effects of `SPlusOnInitializeControl()`, add a line to the “OnDraw” method of `CMyOCXCtrl` by editing the definition of this method in “MyOCXCtrl.h”:

```
void CMyOCXCtrl::OnDraw(
    CDC* pdc, const CRect& rcBounds, const CRect& rcInvalid)
{
    // TODO: Replace the following code with your
    // own drawing code.
    pdc->FillRect(rcBounds,
        CBrush::FromHandle((HBRUSH)GetStockObject(WHITE_BRUSH)));
}
```



```

pdc->Ellipse(rcBounds);

// Display latest value
pdc->DrawText(
  m_sValue, (LPRECT)&rcBounds, DT_CENTER | DT_VCENTER );
}

```

Rebuild the project now to test these changes.

## 7. Test your new control in S-PLUS

To try out your new control in S-PLUS you'll need to create an S-PLUS script which creates properties and displays a dialog. Open S-PLUS and open the script file from SAMPLES/OCX/MyOCX called "MyOCX.SSC". Notice that the script begins by creating three properties, one for the return value from a function and the other two for the parameters of a function. The property for "MyOCX" uses the type "OCX String" and the PROGID for the control we just created:

```

guiCreate("Property",
  name = "MyOCX",
  DialogControl = "OCX String",
  ControlProgId = "MYOCX.MyOCXCtrl.1",
  DialogPrompt = "My &OCX");

```

Run the script "MyOCX.SSC" and you will see a dialog containing an edit field and the MyOCX control you just created. When the dialog appears, the ActiveX control contains the text "Hello" because this is set as the initial value in the S-PLUS script callback function:

```

callbackMyOCXExample <- function(df)
{
  if(IsInitDialogMessage(df)) # Am I called to initialize
    # the properties?
  {
    # Set the initial value of the MyOCX property
    df <- cbSetCurrValue(df,"MyOCX", "\"Hello\"")
  }
  ...
}

```

When you enter a string (use quotes around any string you enter in these dialog fields) in the edit field, the ActiveX control updates to show that string. When you click the OK or Apply buttons in the dialog, you will see the values of both properties printed in a report window.

Summary of steps to support S-PLUS dialogs in ActiveX controls

To summarize the above steps, the list below shows you the tasks necessary to adapt your MFC ActiveX control project to support S-PLUS dialogs:

1. Add S-PLUS dialog support files to your project:

```
OCXUtils.cpp  
OCXUtils.h  
SPDgCInf.cpp  
SPDgCInf.h  
SPlusOCX.cpp  
SPlusOCX.h  
SPlusOCX.idl
```

2. Change the inheritance of your control class from base class `COleControl` to `CSPlusOCX`.
3. Modify your control's ODL (type library definition file) to include `SPlusOCX.idl` sections.
4. Add virtual overrides of key `CSPlusOCX` methods to your control class:

```
virtual long GetSPlusDialogVerticalSize( void );  
virtual long GetSPlusDialogHorizontalSize( void );  
virtual BOOL SPlusOnInitializeControl(const VARIANT  
    FAR& vInitialValue);
```

---

## NEW DIALOG CONTROLS IN S-PLUS 4.5

S-PLUS has a variety of dialog controls that can be used to represent the properties of an object (such as a user-defined function) in a dialog. There are now several new control types in S-PLUS 4.5 that can be used in dialogs you create for user-defined functions.

**Picture** A small rectangle taking up one dialog column which can contain a Windows metafile picture (either Aldus placable or enhanced).

The picture to draw in this control is specified as a string containing either the pathname to the WMF file on disk, or a pathname to a Windows 32-bit DLL followed by the resource name of the metafile picture in this DLL.

**Wide Picture** Same as “**Picture**” except that this control takes up two dialog columns.

**Picture List Box** A scrolling list box control taking up one dialog column which can contain several Windows metafile pictures (either Aldus placable or enhanced).

The list of pictures to draw in this control is specified as a string option list with each element in this option list containing either the pathname to the WMF file on disk, or a pathname to a Windows 32-bit DLL followed by the resource name of the metafile picture in this DLL.

**Wide Picture List Box** Same as “**Picture List Box**” except that this control takes up two dialog columns.

For both the **Picture** and the **Picture List Box** controls, you can specify either a pathname to a Windows metafile on disk or a pathname to a Windows 32-bit DLL and the resource name of the metafile in this DLL to use. The syntax for each of these is specified below:

**Table 7:**

Pathname to Windows metafile	“[pathname]”  Example: “c:\\spluswin\\home\\Meta1.WMF”
DLL Pathname and resource name of metafile	“;[pathname to DLL],[metafile resource name]”  Example: “;c:\\mydll\\mydll.dll, MyMetaFile”  Please note that the leading semicolon is required in this case and the comma is required between the DLL pathname and the name of the metafile resource.

Several example S-PLUS scripts follow which demonstrate how to use these new controls for your own dialogs.

```
# Example script to show how to use a Picture control in a
# dialog in S-PLUS
#

# Define a function for use with this dialog
PictureFn <- function ( GraphToShowEdit )
{
  sPictureShown <- paste(
    sep="", "The graph file: ' ", GraphToShowEdit,
    "' was last shown." )
}

# Create properties for the function
guiCreate("Property", name = "ReturnValue", DialogControl =
  "Invisible" );

guiCreate("Property", name = "GraphToShowEdit",
  DialogControl = "Wide String",
  DialogPrompt = "&Show Graph", UseQuote=T );

# Create the Picture control
guiCreate("Property", name = "Picture1", DialogControl =
  "Picture", DialogPrompt = "&Picture",
  UseQuote=T );
```

```
# Define group property for dialog
guiCreate( "Property", name = "PictureGroup",
           type = "WideGroup",
           DialogPrompt = "Select Picture",
           PropertyList = c( "GraphToShowEdit",
                             "Picture1" ) );

# Function info for the function
guiCreate( "FunctionInfo", Function = "PictureFn",
           DialogHeader = "Picture Control Test",
           PropertyList = c("ReturnValue",
                             "PictureGroup" ),
           ArgumentList = c( "#0 = ReturnValue",
                             "#1 = GraphToShowEdit" ),
           CallbackFunction = "callbackPictureFn",
           Display = "Yes");

# Callback function for this dialog
callbackPictureFn <- function(df)
{
  if(IsInitDialogMessage(df)) # Am I called to initialize
                             # the properties?
  {
    # Set the Picture control to display the Windows
    # metafile referred
    # to by the GraphToShowEdit property
    #
    sPicture1 <- cbGetCurrValue(df, "GraphToShowEdit")
    df <- cbSetCurrValue(df, "Picture1", sPicture1)
  }

  else if( cbIsOkMessage(df)) # Am I called when the Ok
                              # button is pushed?
  {
  }

  else if( cbIsCancelMessage(df)) # Am I called when the
                                  # Cancel button is pushed?
  {
  }

  else if( cbIsApplyMessage(df)) # Am I called when the
                                  #Apply button is pushed?
  {
  }
}
```

```
    }

    else # Am I called when a property value is updated?
    {
        # If the GraphToShowEdit property has been changed,
        # then update the Picture1 picture control to
        # display this metafile.
        #
        if (cbGetActiveProp(df) == "GraphToShowEdit")
        {
            sPicture1 <- cbGetCurrValue(df,
                                     "GraphToShowEdit")
            df <- cbSetCurrValue(df, "Picture1", sPicture1)
        }
    }
    return(df)
}

# Display the dialog
guiDisplayDialog("Function", Name="PictureFn");

# Example script to show how to use a Picture List Box
# control in a dialog in S-PLUS

# Define a function for use with this dialog
PictureListFn <- function (PictureList, GraphSelectedEdit)
{
    sPictureSelected <- paste(
        sep="", "The graph file: '", GraphSelectedEdit,
        "' was selected." )
}

# Create properties for the function
guiCreate("Property", name = "ReturnValue",
        DialogControl = "Invisible" );
guiCreate("Property", name = "GraphSelectedEdit",
        DialogControl = "Wide String", DialogPrompt =
        "&Graph Selected", UseQuote=T );

# Create the picture list box
guiCreate("Property", name = "PictureList",
```

---

```

DialogControl = "Picture List Box",
OptionList = c(
  "c:\\spluswin\\home\\meta1.wmf", # List of
  "c:\\spluswin\\home\\meta2.wmf", #metafiles in
  "c:\\spluswin\\home\\meta3.wmf"), # an option list
DialogPrompt = "&Picture List", UseQuote=T);

# Define group property for dialog
guiCreate( "Property", name = "WidPictureGroup",
  type = "WidGroup", DialogPrompt = "Select Picture",
  PropertyList = c("PictureList", "GraphSelectedEdit") );

# Function info for the function
guiCreate( "FunctionInfo", Function = "PictureListFn",
  DialogHeader = "Picture List Box Control Test",
  PropertyList = c( "ReturnValue", "WidPictureGroup" ),
  ArgumentList = c(
    "#0 = ReturnValue", "#1 = PictureList",
    "#2 = GraphSelectedEdit" ),
  CallbackFunction = "callbackPictureListFn",
  Display = "Yes");

# Callback function for this dialog
callbackPictureListFn <- function(df)
{
  if(IsInitDialogMessage(df)) # Am I called to initialize
    # the properties?
  {
    # Set the GraphSelectedEdit property to the selected
    # metafile pathname in the PictureList property
    #
    sPictureList <- cbGetCurrValue(df, "PictureList")
    df <- cbSetCurrValue(df, "GraphSelectedEdit",
      sPictureList)
  }

  else if( cbIsOkMessage(df)) # Am I called when the Ok
    # button is pushed?
  {

  }

  else if( cbIsCancelMessage(df)) # Am I called when the
    # Cancel button is pushed?
  {

```

```
    }

    else if( cbl$ApplyMessage(df)) # Am I called when the
                                   # Apply button is pushed?
    {
    }

    else # Am I called when a property value is updated?
    {
        # Set the GraphSelectedEdit property to the
        # selected metafile pathname in the PictureList
        # property
        #
        if (cbGetActiveProp(df) == "PictureList")
        {
            sPictureList <- cbGetCurrValue(df,
                                           "PictureList")
            df <- cbSetCurrValue(df, "GraphSelectedEdit",
                                sPictureList)
        }
    }
    return(df)
}

# Show the dialog
guiDisplayDialog("Function", Name="PictureListFn");
```



# NEW SCRIPT WINDOW FEATURES

# 17

---

Automatic Matching of Delimiters	285
Automatic Generation Of Right Braces	285
Automatic Indentation	286
Modifying Script Window Settings	286

Several new features have been added to the Script Window in S-PLUS 4.5. These features are intended to simplify typing S-PLUS functions. Each of these features can be enabled or disabled independently of the others.

## Automatic Matching of Delimiters

S-PLUS automatically matches parentheses ("()"), brackets ("[]"), braces ("{}"), and quotation marks (" ") and ( ' '). For example, whenever you type a right parenthesis (")"), the editor automatically highlights the matching left parenthesis ("("). The behavior is the same for brackets, braces, single quotes ( ' ) and double quotes ( " ). This helps a programmer ensure that matches are as intended.

By default, S-PLUS searches through the entire Script Window to find an automatic match. For large scripts, this can be very time consuming, so you can restrict the search to a specified number of characters.

More precisely, after you type the right parenthesis, the cursor moves automatically to the matching left parenthesis and highlights it for a predetermined length of time (by default 0.5 seconds or 500 milliseconds). The cursor then moves to the the space following the right parenthesis. Any intervening keystrokes are buffered so that no keystrokes are lost if you keep typing while the matching parenthesis is being highlighted. The length of time for highlighting can be changed.

## Automatic Generation Of Right Braces

When automatic generation of right braces ("}") is enabled, pressing Enter after typing a left brace ("{") will result in the automatic insertion of a matching right brace two lines below, and the cursor will be placed on the intervening line.

## Automatic Indentation

When automatic indentation is enabled, the editor automatically indents the bodies of function definitions, `if` statements, `for` statements, and `while` statements. The amount of the indentation is by default 4 spaces, and can be changed. The following sample function illustrates the indentation style that is supported:

```
"test1"<-  
function(x)  
{  
  if(x > 0) {  
    for(i in 1:x) {  
      cat(i, "\n")  
    }  
  }  
  else {  
    i <- - x  
    while(i > 0) {  
      cat(i, "\n")  
      i <- i - 1  
    }  
  }  
}
```

## Modifying Script Window Settings

The default settings of the Script Window can be changed by means of a dialog box accessed by right-clicking in a Script Window and selecting "Property" from the pop-up menu.

To disable any of the following properties, de-select the appropriate check box:

- Auto Match {}, (), [], "" and ''
- Auto Indent
- Auto Insert Right Brace
- Output pane word wrap

To change the Tab Size, enter the number of spaces desired in the appropriate box.

To change the amount of time that matching parentheses etc. are highlighted, change the value in the box labeled, "Match Time (msec)". The value shown is in milliseconds.

---

In order to save the desired settings as defaults for future Script Window sessions, use the "Options:Save Window Size/Properties as Default" menu selection.

To change the number of characters through which S-PLUS will search for an automatic match, enter a value in the Match CharLimit text field. The default value, -1, means to search from the cursor to the top of the file.

The properties of a Script Window can also be accessed from the Object Browser. To do this, first be sure that you are filtering on the Interface Class "Script". (This Interface Class is not included in the filter by default.) Then select "Script" in the left pane of the Object Browser and right-click on the appropriate script in the right pane. Select "Properties" from the pop-up menu.



# INDEX

## Numerics

- 4 Panel Conditioning 22
- 9 Panel Conditioning 22

## A

- Accelerated failure time models 200
- Accelerated testing models 200
- ActiveX Controls in S-Plus dialogs 264
- Adding an ActiveX control to a dialog 264
- add-on modules 13
- Adjusted Means 115
- Agglomerative Hierarchical Clustering 98
- Agglomerative Hierarchical Clustering dialog
  - Model Page 98
  - Plot Page 101
  - Results Page 100
- agnes 101
- analysis of variance table 193
- anova.censorReg 236
- Auto Scale Axes 21
- Automatic Generation Of Right Braces 289
- Automatic Indentation 290
- Automatic Matching of Delimiters 289
- Automating Embedded S-Plus Graphs 258
- Automation Improvements in S-Plus 4.5 245

## B

- Bootstrap Inference dialog 76
  - Jack After Boot page 81
  - Model page 76
  - Options page 77
  - Plot page 80
  - Results page 79

## C

- censor 236
- censorReg 207, 236
  - Accounting for Covariates 210
- censorReg function 207
- censorReg.control 236
- censorReg.distributions 236
- censorReg.fit 236
- clara 93
- ClassName 251
- Clustering In S-Plus 87
- Color 24
- Color Scale Legend 26
  - Hiding 30
  - Showing 30
- Common error conditions when using ActiveX controls in S-PLUS 268
- Comparing Means From Two Samples 143
- Comparing Proportions From Two Samples 148
- Compute Dissimilarities 109
- Computing a Robust Fit 159
- Computing Probabilities and Quantiles 225
- correlation matrix 193
- CreateConditionedPlots 253
- CreateConditionedPlots – SeparateDataGallery 256
- CreateConditionedPlotsGallery 255
- CreateConditionedPlotsSeparateData 254
- CreatePlots 253
- CreatePlotsGallery 255
- Creating HTML Output 111
  - Graphs 114
  - Tables 112
  - Text 113
- Crop Graph to Selected Rectangle 20

## D

daisy 110  
 Designing ActiveX controls that support S-Plus 269  
 Dialog Controls In S-Plus 4.5 263  
 diana 105  
 Display Selected Points 24  
 Dissimilarities 109  
 dissimilarity object 91, 95, 99, 103  
 Divisive Hierarchical Clustering 102  
 Divisive Hierarchical Clustering dialog  
     Model Page 102  
     Plot Page 105  
     Results Page 104

## E

Examples of ActiveX controls included with S-Plus 262  
 Examples of Automation provided with S-PLUS 259  
 Examples Of Using S-plus As An Automation Client Included With S-plus 261  
 Exclude Selected Points 25  
 ExecuteStringResult 251  
 Extract Panel 21  
 Extract Panel/Redraw Graph 21

## F

Factorial Design dialog 60  
     Design Structure 60  
     Names 61  
     Randomization 61  
     Results 61  
 fanny 97  
 formula 236  
 Fuzzy Partitioning 94  
 Fuzzy Partitioning dialog  
     Model Page 94  
     Plot Page 97  
     Results Page 96

## G

GetParameterClasses 252  
 GetSAPIObject 252  
 Graph Options dialog 20  
 Graph Tools Palette 20  
 Graphs dialog  
     Interactive page . 24  
 guiExecuteBuiltIn function 241  
 guiGetAxisLabelsName function 242  
 guiGetAxisName function 242  
 guiGetAxisTitleName function 242  
 guiGetGraphName function 242  
 guiGetGSName function 242  
 guiGetOption function 237  
 guiGetPlotClass function 243  
 guiGetPropertyOptions function 241  
 guiGetRowSelectionExpr function 238  
 guiGetRowSelections function 238  
 guiPlot function 240  
 guiPrintClass function 239  
 guiRemoveContents function 244  
 guiSetOption function 237  
 guiSetRowSelections function 237

## H

Height Multiplier 24  
 Help system  
     On-line Demos 12  
     on-line help 12  
     On-Line Manuals 12  
     training courses 13  
 html.table function 112

## I

Include All Points 25  
 Installation  
     Excel Add-In 34  
     SPSS Add-In 44  
 installing the software 10

**J**

Jackknife Inference dialog 83  
     Model page 83  
     Options page 84  
     Plot page 86  
     Results page 85

**K**

kmeans 89  
 K-Means Clustering 88  
 K-Means Clustering dialog  
     Model Page 88  
     Results Page 89

**L**

Label Point 20  
 Least Squares vs. Robust Fitted Model Objects 160  
 Line Weight Increment 24  
 linear regression 186  
 lm 236  
 lmRobMM function 156

**M**

Meeker, W.Q. 201  
 Method to get and set parameter classes of functions  
 exposed via automation 247  
 MM-estimate 156  
 Modifying Script Window Settings 290  
 modules  
     add-on 13  
 mona 108  
 Monothetic Clustering 106  
 Monothetic Clustering dialog  
     Model Page 106  
     Plot Page 107  
     Results Page 107

**N**

New Automation Methods in S-Plus 4.5 250  
 New Dialog Controls In S-Plus 4.5 283  
 No Conditioning 22

**O**

ObjectContainees 250  
 ObjectContainer 251  
 oil.df data set 158  
 One-Sample Test of Binomial Proportion 146  
 One-Sample Test of Gaussian Mean 140  
 on-line help 12  
 Orthogonal Array Design dialog 62  
     Design Structure 62  
     Randomization 62  
     Results 63

**P**

pam 93  
 Pan Down 21  
 Pan Left 21  
 Pan Right 21  
 Pan Up 21  
 Panels with Varying X and Y Axes 23  
 Parametric Regression For Censored Data 199  
 Parametric Survival dialog 227  
     Model page 227  
     Options page 230  
     Plots page 233  
     Predict page 235  
     Results page 231  
 Partitioning Around Medoids 90  
 Partitioning Around Medoids dialog  
     Model Page 90  
     Plot Page 93  
     Results Page 92  
 Passing Data to Functions via Automation 246  
 PathName 251  
 pftdist 236  
 plot.censorReg 236  
 PLOTDATA.XLS 258

Plots in Separate Panels 22  
 print.censorReg 236

## Q

qftdist 236  
 quantile-quantile plot 195

## R

Recode dialog 64  
     Data 64  
     Values 64  
 Registering an ActiveX control 267  
 Resampling Methods 75  
 Rescale Axes menu 20  
 Reset Auto Scaling 21  
 residual-fit spread plot 195  
 residuals.censorReg 236  
 Return To All Panels 21  
 Robust Linear Regression 153  
 rugplot 195

## S

S 38  
 SAMPLES/OCX 264  
 samples/oleauto 256  
 samples/oleauto/vba 258  
 samples/oleauto/vbembed 258  
 Scale Legend dialog 27  
     Box page 29  
     Labels page 27  
     Position/Size page 30  
     Ticks page 28  
 Select Data 20  
 Select Data dialog 58  
     Existing Data 58  
     New Data 58  
     Show Dialog on Startup 59  
     Source 58  
 Select Tool 20  
 Separate Panels with Varying X Axes 23

Separate Panels with Varying Y Axes 22  
 Set Dimensions dialog 74  
     Data 74  
     Dimensions 74  
 SetParameterClasses 252  
 SetSAPIObject 251  
 setup.exe 10  
 ShowDialogInParent 250  
 ShowDialogInParentModeless 250  
 S-news mailing list 13  
 solve 236  
 Specifying Interval Censored Data 202  
 Specifying the Parametric Family 208  
 Split Data By Group dialog 65  
     Data 65  
     Results 66  
     Splitting Variable 65  
 S-PLUS Excel Add-In  
     Installing 34  
     Selecting data for S-PLUS graphs 40  
     Using the Add-In 38  
 S-PLUS Excel Add-in  
     Removing 37  
 S-Plus Excel Add-In 33  
 S-Press newsletter 14  
 SPSS Add-In 43  
     Installing 44  
     Selecting data for conditioning S-Plus graphs 49  
     Selecting data for S-PLUS graphs 47  
     Using the Add-In 46  
 SPSS Add-in  
     Removing 45  
 Stack Columns dialog 67  
     Data 67  
     Names 68  
     Results 68  
 standard errors 197  
 StatLib 13  
 Style 24  
 Subset dialog 69  
     Data 69  
     Results 70  
 Subset Rows with 21  
 summary.censorReg 236



Survival analysis 200  
Survival data 200  
system requirements 11

## **T**

technical support 14  
The Generalized Kaplan-Meier Estimate 202  
Therneau, Terry 201  
training courses 13  
Transform dialog 71  
    Add to Expression 72  
    Data 71  
Transpose dialog 73  
    Data 73  
    Results 73  
Type III Sum of Squares 115

## **U**

Use Only Selected Points 25

## **V**

VBEMBED.EXE 258

## **W**

Where can the PROGID for the control be found?  
265  
Why only "OCX String"? 268  
Win32s 10  
Windows 3.1 10  
Windows for Workgroups 3.11 10

