

# Package ‘QCAGUI’

October 12, 2009

**Version** 1.3-7

**Date** 2009-10-10

**Title** QCA Graphical User Interface

**Author** This package is 99% based on the Rcmdr package written by John Fox <jfox@mcmaster.ca>. Only some additional menus were adapted for the QCA package by Adrian Dusa <adi@roda.ro>

**Maintainer** Adrian Dusa <adi@sas.unibuc.ro>

**Depends** R (>= 2.1.0), tcltk, grDevices, utils, QCA (>= 0.4-6), MASS

**Suggests** abind, car (>= 1.1-1), foreign, relimp, XML

**LazyLoad** no

**Description** QCAGUI is a graphical user interface (GUI) for the QCA package, derived from R Commander. Because QCA has little to do with statistics, the menus from Rcmdr were stripped down to the very basics. In crisp sets QCA, data is binary therefore it is fairly decent to treat it as categorical (1 - presence; 0 - absence). In order to ease the primary analysis (e.g. tables of frequencies) and the creation of basic graphs, this package activates some menus that are not available in Rcmdr but for factors. Users should be aware, however, that QCAGUI is not a package for statistics; Rcmdr is better for this purpose

**License** GPL (>= 2)

**URL** <http://www.r-project.org>

**Repository** CRAN

**Date/Publication** 2009-10-12 10:22:46

## R topics documented:

QCAGUI-package . . . . .	2
Rcmdr-package . . . . .	3
assignCluster . . . . .	4
big10 . . . . .	5

bin.var . . . . .	6
colPercents . . . . .	7
Commander . . . . .	7
Compute . . . . .	12
dozen12 . . . . .	13
Ebbinghaus . . . . .	13
even11 . . . . .	14
generalizedLinearModel . . . . .	14
hierarchicalCluster . . . . .	15
Hist . . . . .	16
KMeans . . . . .	17
linearModel . . . . .	18
lucky13 . . . . .	19
MI.15 . . . . .	19
mighty14 . . . . .	19
numSummary . . . . .	20
Osa . . . . .	21
partial.cor . . . . .	22
plotMeans . . . . .	22
Plugins . . . . .	24
Rcmdr.sciviews-specific . . . . .	24
Rcmdr.Utilities . . . . .	25
RcmdrPager . . . . .	31
Recode . . . . .	31
reliability . . . . .	32
Rokkan . . . . .	33
stem.leaf . . . . .	34
Stokke . . . . .	36
Yamasaki . . . . .	37

<b>Index</b>	<b>39</b>
--------------	-----------

---

QCAGUI-package      *About the QCAGUI Package*

---

## Description

QCAGUI is a graphical user interface (GUI) for the QCA package, derived from R Commander. Because QCA has little to do with statistics, the menus from Rcmdr were stripped down to the very basics. In crisp sets QCA, data is binary therefore it is fairly decent to treat it as categorical (1 - presence; 0 - absence). In order to ease the primary analysis (e.g. tables of frequencies) and the creation of basic graphs, this package activates some menus that are not available in Rcmdr but for factors. Users should be aware, however, that QCAGUI is not a package for statistics; Rcmdr is better for this purpose

**Details**

Package: QCAGUI  
 Type: Package  
 Version: 1.3-7  
 Date: 2009-10-10  
 License: GPL (>= 2)

**Author(s)**

The package is 99% based on the Rcmdr package written by John Fox <jfox@mcmaster.ca>. Only some additional menus were adapted for the QCA package by Adrian Dusa <adi@sas.unibuc.ro>

---

Rcmdr-package      *R Commander*

---

**Description**

A platform-independent basic-statistics GUI (graphical user interface) for R, based on the tcltk package.

**Details**

Package: Rcmdr  
 Version: 1.3-5  
 Date: 2007.07.30  
 Depends: R (>= 2.1.0), tcltk, grDevices, utils  
 Suggests: abind, car (>= 1.1-1), effects (>= 1.0-7), foreign, grid, lattice, lmtest, MASS, mgcv, multcomp, nlme, nnet, relimp  
 LazyLoad: no  
 License: GPL (>= 2)  
 URL: <http://www.r-project.org>, <http://socserv.socsci.mcmaster.ca/jfox/Misc/Rcmdr/>

**Index:**

Commander	R Commander
Compute	Rcmdr Compute Dialog
Confint	Confidence Intervals for Model Coefficients
Hist	Plot a Histogram
KMeans	K-Means Clustering Using Multiple Random Seeds
Rcmdr.Utilities	Rcmdr Utility Functions
Rcmdr.sciviews-specific	Rcmdr SciViews-specific Functions
RcmdrPager	Pager for Text Files
Recode	Rcmdr Recode Dialog

Scatter3DDialog	Rcmdr 3D Scatterplot Dialog
aboutRcmdr	About the Rcmdr Package
assignCluster	Append a Cluster Membership Variable to a Dataframe
bin.var	Bin a Numeric Variable
colPercents	Row, Column, and Total Percentage Tables
generalizedLinearModel	Rcmdr Generalized Linear Model Dialog
hierarchicalCluster	Rcmdr Hierarchical Clustering Dialog
linearModel	Rcmdr Linear Model Dialog
partial.cor	Partial Correlations
plotMeans	Plot Means for One or Two-Way Layout
reliability	Reliability of a Composite Scale
scatter3d	Three-Dimensional Scatterplots and Point Identification
stem.leaf	Stem-and-Leaf Display

### Translations

The R Commander comes with translations from English into several other languages. I am grateful to the following individuals for preparing these translations: Brazilian Portuguese, Adriano Azevedo Filho; Catalan, Manel Salameo; French, Philippe Grosjean; German: Gerhard Schoen; Indonesian, I Made Tirta; Italian, Stefano Calza; Japanese, Takaharu Araki; Romanian, Adrian Dusa; Russian, Alexey Shipunov; Slovenian, Jaro Lajovic; Spanish, Carlos Enrique Carleos Artime.

### Author(s)

John Fox <jfox@mcmaster.ca>, with contributions from Michael Ash, Theophilus Boye, Stefano Calza, Andy Chang, Philippe Grosjean, Richard Heiberger, G. Jay Kerns, Renaud Lancelot, Matthieu Lesnoff, Martin Maechler, Dan Putler, Miroslav Ristic, and Peter Wolf.

Maintainer: John Fox <jfox@mcmaster.ca>

---

assignCluster

*Append a Cluster Membership Variable to a Dataframe*

---

### Description

Correctly creates a cluster membership variable that can be attached to a dataframe when only a subset of the observations in that dataframe were used to create the clustering solution. NAs are assigned to the observations of the original dataframe not used in creating the clustering solution.

### Usage

```
assignCluster(clusterData, origData, clusterVec)
```

**Arguments**

- `clusterData` The data matrix used in the clustering solution. The data matrix may have only a subset of the observations contained in the original dataframe.
- `origData` The original dataframe from which the data used in the clustering solution were taken.
- `clusterVec` An integer variable containing the cluster membership assignments for the observations used in creating the clustering solution. This vector can be created using `cutree` for clustering solutions generated by `hclust` or the `cluster` component of a list object created by `kmeans` or `KMeans`.

**Value**

A factor (with integer labels) that indicate the cluster assignment for each observation, with an NA value given to observations not used in the clustering solution.

**Author(s)**

Dan Putler

**See Also**

[hclust](#), [cutree](#), [kmeans](#), [KMeans](#)

**Examples**

```
data(USArrests)
USArrkm3 <- KMeans(USArrests[USArrests$UrbanPop<66, ], centers=3)
assignCluster(USArrests[USArrests$UrbanPop<66, ], USArrests, USArrkm3$cluster)
```

---

big10

*Random data with 10 conditions and 1 outcome*

---

**Description**

The data was randomly generated using `set.seed(10)`

**Usage**

```
data(big10)
```

---

`bin.var`*Bin a Numeric Variable*

---

**Description**

Create a factor dissecting the range of a numeric variable into bins of equal width, (roughly) equal frequency, or at "natural" cut points. The `cut` function is used to create the factor.

**Usage**

```
bin.var(x, bins = 4, method = c("intervals", "proportions", "natural"),
        labels = FALSE)
```

**Arguments**

<code>x</code>	numeric variable to be binned.
<code>bins</code>	number of bins.
<code>method</code>	one of "intervals" for equal-width bins; "proportions" for equal-count bins; "natural" for cut points between bins to be determined by a k-means clustering.
<code>labels</code>	if FALSE, numeric labels will be used for the factor levels; if NULL, the cut points are used to define labels; otherwise a character vector of level names.

**Value**

A factor.

**Author(s)**

Dan Putler, slightly modified by John Fox ([jfox@mcmaster.ca](mailto:jfox@mcmaster.ca)) with the original author's permission.

**See Also**

[cut](#), [kmeans](#).

**Examples**

```
summary(bin.var(rnorm(100), method="prop", labels=letters[1:4]))
```

---

`colPercents`*Row, Column, and Total Percentage Tables*

---

**Description**

Percentage a matrix or higher-dimensional array of frequency counts by rows, columns, or total frequency.

**Usage**

```
colPercents(tab, digits=1)
rowPercents(tab, digits=1)
totPercents(tab, digits=1)
```

**Arguments**

`tab` a matrix or higher-dimensional array of frequency counts.  
`digits` number of places to the right of the decimal place for percentages.

**Value**

Returns an array of the same size and shape as `tab` percentaged by rows or columns, plus rows or columns of totals and counts, or by the table total.

**Author(s)**

John Fox <jfox@mcmaster.ca>

---

`Commander`*R Commander*

---

**Description**

Start the R Commander GUI (graphical user interface)

**Usage**

```
Commander()
```

## Details

### Getting Started

The default R Commander interface consists of (from top to bottom) a menu bar, a toolbar, a script window, an output window, and a messages window.

Commands to read, write, transform, and analyze data are entered using the menus in the menu bar at the top of the *Commander* window. Most menu items lead to dialog boxes requesting further specification. I suggest that you explore the menus to see what is available.

Below the menu bar is a toolbar with (from left to right) an information field displaying the name of the active data set; buttons for editing and displaying the active data set; and an information field showing the active statistical model. There is also a *Submit* button for re-executing commands in the script window. The information fields for the active data set and active model are actually buttons that can be used to select the active data set and model from among, respectively, data frames or suitable model objects in memory.

Almost all commands require an active data set. When the Commander starts, there is no active data set, as indicated in the data set information field. A data set becomes the active data set when it is read into memory from an R package or imported from a text file, SPSS data set, Minitab data set, STATA data set, or an Excel, Access, or dBase data set. In addition, the active data set can be selected from among R data frames resident in memory. You can therefore switch among data sets during a session.

By default, commands are logged to the script window (the initially empty text window immediately below the toolbar); commands and output appear in the output window (the initially empty text window below the script window); and the active data set is attached to the search path. To alter these and other defaults, see the information below on configuration.

Some `Rcmdr` dialogs (those in the *Statistics -> Fit models* menu) produce linear, generalized linear, or other models. When a model is fit, it becomes the active model, as indicated in the information field in the R Commander toolbar. Items in the *Models* menu apply to the active model. Initially, there is no active model. If there are several models in memory, you can select the active model from among them.

If command logging is turned on, R commands that are generated from the menus and dialog boxes are entered into the script window in the Commander. You can edit these commands in the normal manner and can also type new commands into the script window. Individual commands can be continued over more than one line, the several lines of a multi-line command must be submitted simultaneously. (It is not necessary, as in earlier versions of the R Commander, to begin continuation lines with white space.) The contents of the script window can be saved during or at the end of the session, and a saved script can be loaded into the script window. The contents of the output window can also be edited or saved to a text file.

To re-execute a command or set of commands, select the lines to be executed using the mouse and press the *Submit* button at the right of the toolbar (or *Control-R*, for "run"). If no text is selected, the *Submit* button (or *Control-R*) submits the line containing the text-insertion cursor. Note that an error will be generated if the submitted command or commands are incomplete.

Pressing *Control-F* brings up a find-text dialog box (which can also be accessed via *Edit -> Find*) to search for text in the script window or the output window. Edit functions such as search are performed in the script window unless you first click in the output window to make it the active window.

Pressing *Control-S* will save the script or output window.

Pressing *Control-A* selects all of the text in the script or output window.

Right-clicking the mouse (clicking button 3 on a three-button mouse) in the script or output window brings up a "context" menu with the *Edit*-menu items, plus (in the script window) a *Submit* item.

When you execute commands from the *Commander* window, you must ensure that the sequence of commands is logical. For example, it makes no sense to fit a statistical model to a data set that has not been read into memory.

Pressing a letter key (e.g., "a") in a list box will scroll the list box to bring the next entry starting with that letter to the top of the box.

Exit from the Commander via the *File -> Exit* menu or by closing the *Commander* window.

### Customization and Configuration

Configuration files reside in the `etc` subdirectory of the package, or in the locations given by the `etc` and `etcMenus` options (see below).

The `Rcmdr` menus can be customized by editing the file `Rcmdr-menus.txt`.

Some functions (e.g., `hist`) that do not normally create visible printed output when executed from the *R Console* command prompt will do so — unless prevented — when executed from the *Commander* script window. Such output can be suppressed by listing the names of these functions in the `log-exceptions.txt` file.

You can add R code to the package, e.g., for creating additional dialogs, by placing files with file type `.R` in the `etc` directory, also editing `Rcmdr-menus.txt` to provide additional menus, sub-menus, or menu-items. A demo addition is provided in the file `BoxCox.demo`. To activate the demo, rename this file to `BoxCox.R`, and uncomment the corresponding menu line in `Rcmdr-menus.txt`. Alternatively, you can edit the source package and recompile it.

A number of functions are provided to assist in writing dialogs, and `Rcmdr` state information is stored in a separate environment. See `help("Rcmdr.Utilities")` and the manual supplied in the `doc` directory of the `Rcmdr` package for more information.

In addition, several features are controlled by run-time options, set via the `options("Rcmdr")` command. These options should be set before the package is loaded. If the options are unset, which is the usual situation, defaults are used. Specify options as a list of `name=value` pairs. You can set none, one, several, or all options. The available options are as follows:

**attach.data.set** if TRUE (the default is FALSE), the active data set is attached to the search path.

**check.packages** if TRUE (the default), on start-up, the presence of all of the `Rcmdr` recommended packages will be checked, and if any are absent, the `Rcmdr` will offer to install them.

**command.text.color** Color for commands in the output window; the default is "red".

**console.output** If TRUE, output is directed to the *R Console*, and the *R Commander* output window is not displayed. The default is FALSE.

**contrasts** Serves the same function as the general `contrasts` option; the default is `c("contr.Treatment", "contr.poly")`. When the Commander exits, the `contrasts` option is returned to its pre-existing value. Note that `contr.Treatment` is from the `car` package.

**crisp.dialogs** If TRUE, dialogs should appear on the screen fully drawn, rather than built up widget by widget. This option should affect the Windows version of R only, but should in any event be harmless. The default is TRUE under Windows for R versions 2.1.1 and above, and

FALSE otherwise. If you're working on Windows and encounter increased stability problems, trying setting this option to FALSE.

**default.font** The default font, as an X11 font specification, given in a character string. If specified, this value takes precedence over the default font size (below). This option is only for non-Windows systems.

**default.font.size** The size, in points, of the default font. The default is 10 for Windows systems and 12 for other systems Unless otherwise specified (see the previous item), the default font is `"*helvetica-medium-r-normal-*-*xx*"`, where `xx` is the default font size. This option is only for non-Windows systems.

**double.click** Set to TRUE if you want a double-click of the left mouse button to press the default button in all dialogs. The default is FALSE.

**error.text.color** Color for error messages; the default is "red".

**etc** Set to the path of the directory containing the Rcmdr configuration files; defaults to the `etc` subdirectory of the installed Rcmdr package.

**grab.focus** Set to TRUE for the current Tk window to "grab" the focus — that is, to prevent the focus from being changed to another Tk window. On some systems, grabbing the focus in this manner apparently causes problems. The default is TRUE. If you experience focus problems, try setting this option to FALSE.

**load.at.startup** A character vector of names of packages to be loaded when the Rcmdr package is loaded; the default is to load only the `car` package. Other required packages will be loaded as needed. If it is available, the `car` package will be loaded at when the Commander starts in any event.

**log.commands** If TRUE (the default), commands are echoed to the script window; if FALSE, the script window is not displayed.

**log.font.size** The font size, in points, to be used in the script window, in the output window, in recode dialogs, and in compute expressions — that is, where a monospaced font is used. The default is 10 for Windows systems and 12 for other systems.

**log.height** The height of the script window, in lines. The default is 10. Setting `log.height` to 0 has the same effect as setting `log.commands` to FALSE.

**log.text.color** Color for text in the script window; the default is "black".

**log.width** The width of the script and output windows, in characters. The default is 80.

**multiple.select.mode** Affects the way multiple variables are selected in variable-list boxes. If set to "extended" (the default), left-clicking on a variable selects it and deselects any other variables that are selected; Control-left-click toggles the selection (and may be used to select additional variables); Shift-left-click extends the selection. This is the standard Windows convention. If set to "multiple", left-clicking toggles the selection of a variable and may be used to select more than one variable. This is the behaviour in the Rcmdr prior to version 1.9-10.

**output.height** The height of the output window, in lines. The default is twice the height of the script window, or 20 if the script window is suppressed. Setting `output.height` to 0 has the same effect as setting `console.output` to TRUE.

**output.text.color** Color for output in the output window; the default is "blue".

**placement** Placement of the *R Commander* window, in pixels; the default is `"-40+20"`, which puts the window near the upper-right corner of the screen.

- plugins** A character vector giving the names of Rcmdr plug-in packages to load when the Commander starts up. Plug-in packages can also be loaded from the *Tools -> Load Rcmdr plug-in(s)* menu. See [Plugins](#).
- suppress.menus** if TRUE, the Commander menu bar and tool bar are suppressed, allowing another program (such as Excel) to take over these functions. The default (of course) is FALSE.
- suppress.X11.warnings** On (some?) Linux systems, multiple X11 warnings are generated by Rcmdr commands after a graphics-device window has been opened. Set this option to TRUE (the default when running interactively under X11 prior to R version 2.4.0) to suppress reporting of these warnings. An undesirable side effect is that then *all* warnings and error messages are intercepted by the Rcmdr, even those for commands entered at the R command prompt. Messages produced by such commands will be printed in the Commander Messages window after the next Rcmdr-generated command. Some X11 warnings may be printed when you exit from the Commander. This problem only applies to R versions before 2.4.0, and the default value of the option is set accordingly.
- retain.messages** If TRUE (the default is FALSE), the contents of the message window are not erased between messages. In any event, a "NOTE" message will not erase a preceding "WARNING" or "ERROR".
- RExcelSupport** If TRUE (the default is FALSE), menus and output are handled by Excel.
- scale.factor** A scaling factor to be applied to all Tk elements, such as fonts. This works well only in Windows. The default is NULL.
- showData.threshold** If the number of variables in the active data set exceeds this value (default, 100), then `edit()` rather than `showData()` is used to display the data set. A disadvantage is that control doesn't return to the Commander until the edit window is closed. The reason for the option is that `showData()` is very slow when the number of variables is large; setting the threshold to 0 suppresses the use of `showData` altogether.
- show.edit.button** Set to TRUE (the default) if you want an *Edit* button in the Commander window, permitting you to edit the active data set. Windows users may wish to set this option to FALSE to suppress the *Edit* button because changing variable names in the data editor can cause R to crash (though I believe that this problem has been solved).
- sort.names** Set to TRUE (the default) if you want variable names to be sorted alphabetically in variable lists.
- tkwait** This option addresses a problem that, to my knowledge, is rare, and may occur on some non-Windows systems. If the Commander causes R to hang, then set the `tkwait` option to TRUE; otherwise set the option to FALSE or ignore it. An undesirable side effect of setting the `tkwait` option to TRUE is that the R session command prompt is suppressed until the Commander exits. One can still enter commands via the script window, however. In particular, there is no reason to use this option under Windows, and it should not be used with the Windows R GUI with buffered output when output is directed to the R console.
- use.rgl** If TRUE (the default), the `rgl` package will be loaded if it is present in an accessible library; if FALSE, the `rgl` package will be ignored even if it is available. The `rgl` package can sometimes cause problems when running R under X11.
- warning.text.color** Color for warning messages; the default is "darkgreen".

Many options can also be set via the *File -> Options* menu, which will restart the Commander after options are set.

**Warning**

The R Commander Script window does not provide a true console to R, and has certain limitations. I don't recommend using the R Commander for serious programming or for data analysis that relies primarily on scripts — use a programming editor instead. For example, compound R statements enclosed in braces " { } ", including in function definitions, will not be parsed and executed correctly, even if lines after the first are indented. You can execute compound statements from the Script window by separating commands within braces by semicolons.

**Known Problem**

Occasionally, under Windows, after typing some text into a dialog box (e.g., a subsetting expression in the Subset Data Set dialog), buttons in the dialog (e.g., the OK button) will have no effect when they are pressed. Clicking anywhere inside or outside of the dialog box should restore the function of the buttons. As far as I have been able to ascertain, this is a problem with Tcl/Tk for Windows.

**Note**

This version may be compatible with SciViews, which currently runs only under Windows systems: <http://www.sciviews.org/SciViews-R>; see [Rcmdr.sciviews-specific](#). Under Windows, the `Rcmdr` package can also be run under the Rgui in SDI (single-document interface) mode, or under `rterm.exe`; you might experience problems running the `Rcmdr` under ESS with NTEmacs or XEmacs.

**Author(s)**

John Fox (jfox@mcmaster.ca)

**See Also**

[Plugins](#)

**Examples**

```
options(Rcmdr=list(log.font.size=12, contrasts=c("contr.Sum", "contr.poly")))
```

---

Compute

*Rcmdr Compute Dialog*

---

**Description**

The compute dialog is used to compute new variables.

**Details**

The name of the new variable must be a valid R object name (consisting only of upper and lower-case letters, numerals, and periods, and not starting with a numeral).

Enter an R expression in the box at the right. The expression is evaluated using the active data set. You can double-click in the variable-list box to enter variable names in the expression. The expression must evaluate to a valid variable, which is added to the active data set.

**Author(s)**

John Fox (jfox@mcmaster.ca)

**See Also**

[Arithmetic](#)

---

dozen12

*Random data with 12 conditions and 1 outcome*

---

**Description**

The data was randomly generated using `set.seed(12)`

**Usage**

```
data(dozen12)
```

---

Ebbinghaus

*Union Growth and Decline in Western Europe 1950-95*

---

**Description**

The **Ebbinghaus** data frame has 13 rows and 10 columns.

During the early post-war period, Western trade union movements grew in membership and achieved an institutionalized role in industrial relations and politics. However, during the last decades, many trade unions have seen their membership decline as they came increasingly under pressures due to the social, economic and political changes. This article reviews the main structural, cyclical and institutional factors explaining union growth and decline. Concentrating on Western Europe, the empirical analysis compares cross-national union density data for 13 countries over the first period (1950-75) and for 16 countries over the second, "crisis" period (1975-95)

**Usage**

```
data(Ebbinghaus)
```

**Format**

The dataset contains the following columns:

OUT	union decline (outcome variable)
GHENT	country with "Ghent-system" (union led unemployment insurance)
WORKACCESS	workplace access for unions
NEOCORP	neo-corporatist institutionalization of unions
CLOSHOP	"closed-shop" tradition
SOCDEM	government participation of Social-Democratic / Socialist party
INDUS	share of industry in dependent unemployment
PUBLIC	share of public sector development employment
UNEMPL	average unemployment rate
INFLA	average inflation rate

**Source**

<http://www.compass.org>

**References**

Ebbinghaus, Bernhard and Visser, Jelle 2000 *Trade Unions in Western Europe since 1945*. In Flora, P. and Rothenbacher, Kraus F. (eds.) *The Societies of Europe Series*, London: Macmillan Reference / Palgrave, March 2000 / New York: Grove's Dictionaries / Palgrave, July 2000 [xxii, 808 pp. and CD-ROM]

---

even11

*Random data with 11 conditions and 1 outcome*

---

**Description**

The data was randomly generated using `set.seed(11)`

**Usage**

```
data(even11)
```

---

generalizedLinearModel

*Rcmdr Generalized Linear Model Dialog*

---

**Description**

This dialog is used to specify a generalized linear model to be fit by the `glm` function.

### Details

The left model-formula box specifies the response variable to be used in the model; it may be a variable name or an expression evaluating to the response variable, such as `working == "Fulltime"`.

The right model-formula box specifies the right-hand (i.e., predictor) side of the model. See [glm](#) for details.

You can type directly in the model formula boxes. Alternatively, double-clicking the left mouse button on a variable in the variable-list transfers it to the left-hand side of the model (if it is empty) or to the right-hand side. Factors are indicated in the variable list; all other variables are numeric. You can also enter operators and parentheses using the buttons above the formula.

Double-click the left mouse button to select a family in the "Family" box and the corresponding permissible link functions appear in the "Link function" box to the right. Initially, the canonical link for the family is selected. See [family](#) for details.

Specifying a subset expression allows you to fit the model to a subset of observations in the active data set. For example, assuming that `gender` is a variable in the active data set, entering `gender == "Male"` would restrict the model to males.

If the active model is a generalized linear model, and the active data set has not changed, then the initial values of the left-hand-side, right-hand-side, family, link, and subset fields are retained from the active model.

### Author(s)

John Fox ([jfox@mcmaster.ca](mailto:jfox@mcmaster.ca))

### See Also

[glm](#), [family](#), [Comparison](#)

---

hierarchicalCluster

*Rcmdr Hierarchical Clustering Dialog*

---

### Description

This dialog is used to specify a hierarchical cluster analysis solution using [hclust](#), with the distance matrix calculated using [dist](#).

### Details

Enter a name for the hierarchical clustering solution to be created if you want to retain more than one solution. The solution name must be a valid R object name (consisting only of upper- and lower-case letters, numerals, and periods, and not starting with a number).

Select the variables to be included in the solution using the variable selection box on the left side of the dialog box. A non-contiguous set of variables can be selected by pressing your control key (ctrl) while selecting variables.

Specifying a subset expression (the field below the variable selection box) allows you to obtain a clustering solution for a subset of observations in the active data set. For example, assuming that `gender` is a variable in the active data set, entering `gender == "Male"` would restrict the solution to males.

Select a clustering method and a distance measure if you are working with raw data. There is often a relationship between the selection of these two items. For example, squared-euclidian distance is appropriate for Ward's method of cluster analysis. If your data *is* a distance matrix, then select "No Transformation" as the distance measure.

The "Plot Dendrogram" option results in the dendrogram of the solution being display by using the `plot` function.

**Author(s)**

Dan Putler

**See Also**

[hclust](#), [dist](#)

---

Hist

*Plot a Histogram*

---

**Description**

This function is a wrapper for the `hist` function in the `base` package, permitting percentage scaling of the vertical axis in addition to frequency and density scaling.

**Usage**

```
Hist(x, scale = c("frequency", "percent", "density"), ...)
```

**Arguments**

<code>x</code>	a vector of values for which a histogram is to be plotted.
<code>scale</code>	the scaling of the vertical axis: "frequency" (the default), "percent", or "density".
<code>...</code>	arguments to be passed to <code>hist</code> .

**Value**

This function returns `NULL`, and is called for its side effect — plotting a histogram.

**Author(s)**

John Fox ([jfox@mcmaster.ca](mailto:jfox@mcmaster.ca))

**See Also**[hist](#)**Examples**

```
data(Prestige)
Hist(Prestige$income, scale="percent")
```

KMeans

*K-Means Clustering Using Multiple Random Seeds***Description**

Finds a number of k-means clustering solutions using R's `kmeans` function, and selects as the final solution the one that has the minimum total within-cluster sum of squared distances.

**Usage**

```
KMeans(x, centers, iter.max=10, num.seeds=10)
```

**Arguments**

<code>x</code>	A numeric matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a dataframe with all numeric columns).
<code>centers</code>	The number of clusters in the solution.
<code>iter.max</code>	The maximum number of iterations allowed.
<code>num.seeds</code>	The number of different starting random seeds to use. Each random seed results in a different k-means solution.

**Value**

A list with components:

<code>cluster</code>	A vector of integers indicating the cluster to which each point is allocated.
<code>centers</code>	A matrix of cluster centres (centroids).
<code>withinss</code>	The within-cluster sum of squares for each cluster.
<code>tot.withinss</code>	The within-cluster sum of squares summed across clusters.
<code>betweenss</code>	The between-cluster sum of squared distances.
<code>size</code>	The number of points in each cluster.

**Author(s)**

Dan Putler

**See Also**[kmeans](#)**Examples**

```
data(USArrests)
KMeans(USArrests, centers=3, iter.max=5, num.seeds=5)
```

---

`linearModel`*Rcmdr Linear Model Dialog*

---

**Description**

This dialog is used to specify a linear model to be fit by the [lm](#) function.

**Details**

The left model-formula box specifies the response variable to be used in the model; it may be a variable name or an expression evaluating to the response variable, such as `log(income)`.

The right model-formula box specifies the right-hand (i.e., predictor) side of the model. See [lm](#) for details.

You can type directly in the model formula boxes. Alternatively, double-clicking the left mouse button on a variable in the variable-list transfers it to the left-hand side of the model (if it is empty) or to the right-hand side. You can also enter operators and parentheses using the buttons above the formula.

Specifying a subset expression allows you to fit the model to a subset of observations in the active data set. For example, assuming that `gender` is a variable in the active data set, entering `gender == "Male"` would restrict the model to males.

If the active model is a linear model and the active data set has not changed, then the initial values of the left-hand-side, right-hand-side, and subset fields are retained from the previous model.

**Author(s)**

John Fox <jfox@mcmaster.ca>

**See Also**[lm](#), [Comparison](#)

---

`lucky13`*Random data with 13 conditions and 1 outcome*

---

**Description**

The data was randomly generated using `set.seed(13)`

**Usage**

```
data(lucky13)
```

---

`MI.15`*Random data with 15 conditions and 1 outcome*

---

**Description**

The data was randomly generated using `set.seed(15)` "MI" stands for Mission Impossible

**Usage**

```
data(MI.15)
```

---

`mighty14`*Random data with 14 conditions and 1 outcome*

---

**Description**

The data was randomly generated using `set.seed(14)`

**Usage**

```
data(mighty14)
```

---

`numSummary`*Mean, Standard Deviation, and Quantiles for Numeric Variables*

---

### Description

`numSummary` creates neatly formatted tables of means, standard deviations, and quantiles of numeric variables.

### Usage

```
numSummary(data, statistics=c("mean", "sd", "quantiles"),
           quantiles=c(0, .25, .5, .75, 1), groups)
```

```
## S3 method for class 'numSummary':
print(x, ...)
```

### Arguments

<code>data</code>	a numeric vector, matrix, or data frame.
<code>statistics</code>	any of "mean", "sd", or "quantiles", defaulting to all three.
<code>quantiles</code>	quantiles to report; default is <code>c(0, 0.25, 0.5, 0.75, 1)</code> .
<code>groups</code>	optional variable, typically a factor, to be used to partition the data.
<code>x</code>	object of class "numSummary" to print.
<code>...</code>	arguments to pass down from the print method.

### Value

`numSummary` returns an object of class "numSummary" containing the table of statistics to be reported along with information on missing data, if there are any.

### Author(s)

John Fox ([jfox@mcmaster.ca](mailto:jfox@mcmaster.ca))

### See Also

[mean](#), [sd](#), [quantile](#).

### Examples

```
library(car)
Prestige[1, "income"] <- NA
numSummary(Prestige[,c("income", "education")])
numSummary(Prestige[,c("income", "education")], groups=Prestige$type)
remove(Prestige)
```

**Description**

The **Osa** data frame has 24 rows and 6 columns.

This data is drawn from a study which analyzes twenty-four cases of occurrence/non-occurrence of mobilization in non-democratic states to determine conditions of political opportunity in high-risk authoritarian contexts. Political opportunity is sensitive to conditions created by divided elites, changes in repression, media access, influential allies, and social networks

**Usage**

```
data(Osa)
```

**Format**

The dataset contains the following columns:

DYNA	Dynamics of repression, coded 1 for evidence of increase in long-term and/or short-term state repression, 0 for evidence of decrease
ACCESS	Media access - coded 1 if the public information flow concerning a particular event of popular mobilization is controlled by the political authorities (via censorship or ban on foreign media), 0 if there was evidence of sustained relaxation of state censorship or substantive presence of foreign and/or underground media
INFLU	Influential allies - coded 1 for presence of domestic or foreign politically influential groups supporting popular mobilization; 0 for absence of such organizational support
ELITE	Division of elites, coded 1 - evidence of competing factions within the ruling elites; 0 - relatively unified ruling group
SOCIAL	Social networks - coded 1 if mobilization resulted from the activity of interconnected groups, 0 if organizational and/or individual ties were severely destroyed or impeded to emerge by the state
OUT	Social mobilization, coded 1 for major episodes of sustained collective action opposing state policies by participants drawn from nonelite or repressed segments of society in non-democratic regimes, 0 - non-mobilization

**Source**

<http://www.compass.org>

**References**

Osa, Maryjane and Corduneanu-Huci, Cristina 2003 *Running Uphill: Political Opportunity in Non-democracies*, *Comparative Sociology* 2(4), pp. 605-629(25)

---

`partial.cor`*Partial Correlations*

---

**Description**

Computes a matrix of partial correlations between each pair of variables controlling for the others.

**Usage**

```
partial.cor(X, ...)
```

**Arguments**

<code>X</code>	data matrix.
<code>...</code>	arguments to be passed to <code>cor</code> .

**Value**

Returns a matrix of partial correlations.

**Author(s)**

John Fox <jfox@mcmaster.ca>

**See Also**

[cor](#)

**Examples**

```
data(DavisThin)
partial.cor(DavisThin)
```

---

`plotMeans`*Plot Means for One or Two-Way Layout*

---

**Description**

Plots cell means for a numeric variable in each category of a factor or in each combination of categories of two factors, optionally along with error bars based on cell standard errors or standard deviations.

**Usage**

```
plotMeans(response, factor1, factor2,  
          error.bars = c("se", "sd", "conf.int", "none"), level=0.95,  
          xlab = deparse(substitute(factor1)),  
          ylab = paste("mean of", deparse(substitute(response))),  
          legend.lab = deparse(substitute(factor2)), main = "Plot of Means",  
          pch = 1:n.levs.2, lty = 1:n.levs.2, col = palette())
```

**Arguments**

response	Numeric variable for which means are to be computed.
factor1	Factor defining horizontal axis of the plot.
factor2	If present, factor defining profiles of means
error.bars	If "se", the default, error bars around means give plus or minus one standard error of the mean; if "sd", error bars give plus or minus one standard deviation; if "conf.int", error bars give a confidence interval around each mean; if "none", error bars are suppressed.
level	level of confidence for confidence intervals; default is .95
xlab	Label for horizontal axis.
ylab	Label for vertical axis.
legend.lab	Label for legend.
main	Label for the graph.
pch	Plotting characters for profiles of means.
lty	Line types for profiles of means.
col	Colours for profiles of means

**Value**

The function invisibly returns NULL.

**Author(s)**

John Fox <jfox@mcmaster.ca>

**See Also**

[interaction.plot](#)

## Description

Plug-ins are R packages that extend the R Command interface.

## Details

An R Commander plug-in is an ordinary R package that (1) provides extensions to the R Commander menus is a file named `menus.txt` located in the package's `etc` directory; (2) provides call-back functions required by these menus; and (3) in optional `Log-Exceptions:` and `Models:` fields in the package's `DESCRIPTION` file, augments respectively the list of functions for which printed output is suppressed and the list of model objects recognized by the R Commander. The menus provided by a plug-in package are merged with the standard Commander menus.

Plug-in packages given in the R Commander `plugins` option (see [Commander](#)) are automatically loaded when the Commander starts up. Plug-in packages may also be loaded via the Commander *Tools -> Load Rcmdr plug-in(s)* menu; a restart of the Commander is required to install the new menus. Finally, loading a plug-in package when the **Rcmdr** is not loaded will load the **Rcmdr** and activate the plug-in.

An illustrative R Commander plug-in package, **RcmdrPlugin.TeachingDemos**, is available on CRAN.

## See Also

[Commander](#)

## Description

These functions provide compatibility with SciViews (<http://www.sciviews.org>). Thanks to them, Rcmdr is totally integrated into SciViews Insider. In this environment, the main 'R Commander' window is replaced by an 'R Commander menu' and log files are replaced by special R code editing windows with syntax highlighting. Most of these functions are not intended for direct use.

**Usage**

```
is.SciViews()  
is.SciViews.TclTk()  
svlogger(command)  
optionLogCommand()  
optionAttachDataSet()  
optionSortVariables()  
refreshStatus()
```

**Arguments**

`command` a character string that evaluates to an R command.

**Details**

The functions `is.SciViews` tests if R is running under SciViews. If not, most of the other SciViews-specific functions do nothing. `is.SciViews.TclTk` test if the SciViews client communicates with R through Tcl/Tk (otherwise, it probably uses SciViews plugs). The function `svlogger` is similar to `logger`, but it records Rcmdr commands in the specific SciViews R script window and in the SciViews command history, instead of the log window and the default R command history. `optionLogCommand`, `optionAttachDataSet` and `optionSortVariables` allow to change the command logging, automatic attachment of the active data set and sorting of variable names (equivalent options than those accessible by check boxes in the 'R Commander' window of Rcmdr outside of SciViews, or in the Options dialog box). In SciViews insider, the state of these options, as well as the names of the active data set and model are displayed in the status bar. `refreshStatus` make sure that this information in the status bar is updated according to the current internal state of Rcmdr.

**Author(s)**

Philippe Grosjean (phgrosjean@sciviews.org)

---

Rcmdr.Utilities      *Rcmdr Utility Functions*

---

**Description**

These functions support writing additions to the Rcmdr package. Additional R code can be placed in files with file type `.R` in the `etc` subdirectory of the package. Add menus, submenus, and menu items by editing the file `Rcmdr-menus.txt` in the same directory.

**Usage**

```
activateMenus()  
activeDataSet(dsname, flushModel=TRUE)  
ActiveDataSet(name)  
activeDataSetP()
```

```

activeModel(model)
ActiveModel(name)
activeModelP()
checkActiveDataSet()
checkActiveModel()
checkBoxes(window=top, frame, boxes, initialValues=NULL, labels) # macro
checkClass(object, class, message=NULL) # macro
checkFactors(n=1)
checkMethod(generic, object, message=NULL, default=FALSE, strict=FALSE,
            reportError=TRUE) # macro
checkNumeric(n=1)
checkReplace(name, type=gettextRcmdr("Variable"))
checkTwoLevelFactors(n=1)
checkVariables(n=1)
closeCommander(ask=TRUE, ask.save=ask)
closeDialog(window, release=TRUE) # macro
CommanderWindow()
dataSetsP()
defmacro(..., expr)
dialogSuffix(window=top, onOK=onOK, rows=1, columns=1, focus=top, bindReturn=TRUE,
            preventGrabFocus=FALSE, preventDoubleClick=FALSE, preventCrisp=FALSE) # macro
doItAndPrint(command, log=TRUE)
errorCondition(window=top, recall=NULL, message, model=FALSE) # macro
exists.method(generic, object, default=TRUE, strict=FALSE)
Factors(names)
factorsP(n=1)
## S3 method for class 'listbox':
getFrame(object)
## S3 method for class 'listbox':
getSelection(object)
getRcmdr(x, mode="any")
gettextRcmdr(...)
glmP()
GrabFocus(value)
groupsBox(recall=NULL, label=gettextRcmdr("Plot by:"),
          initialLabel=gettextRcmdr("Plot by groups"),
          plotLinesByGroup=FALSE, positionLegend=FALSE,
          plotLinesByGroupsText=gettextRcmdr("Plot lines by group")) # macro
groupsLabel(frame=top, groupsBox=groupsBox, colspan=1) # macro
hclustSolutionsP()
initializeDialog(window=top, title="", offset=10, preventCrisp=FALSE) # macro
is.valid.name(x)
justDoIt(command)
listAllModels(envir=.GlobalEnv, ...)
listDataSets(envir=.GlobalEnv, ...)
listFactors(dataSet=ActiveDataSet())
listGeneralizedLinearModels(envir=.GlobalEnv, ...)
listLinearModels(envir=.GlobalEnv, ...)

```

```

listMultinomialLogitModels(envir=.GlobalEnv, ...)
listNumeric(dataSet=ActiveDataSet())
listPlugins(loaded=FALSE)
listProportionalOddsModels(envir=.GlobalEnv, ...)
listTwoLevelFactors(dataSet=ActiveDataSet())
listVariables(dataSet=ActiveDataSet())
lmP()
logger(command)
LogWindow()
Message(message, type=c("note", "error", "warning"))
MessagesWindow()
modelFormula(frame=top, hasLhs=TRUE) # macro
modelsP(n=1)
Numeric(names)
numericP(n=1)
OKCancelHelp(window=top, helpSubject=NULL, model=FALSE) # macro
OutputWindow()
packageAvailable(name)
putRcmdr(x, value)
radioButtons(window=top, name, buttons, values=NULL, initialValue=..values[1],
             labels, title) # macro
RcmdrTclSet(name, value)
RcmdrTkmessageBox(message, icon=c("info", "question", "warning",
                                "error"), type=c("okcancel", "yesno", "ok"), default, title="")
rglLoaded()
subOKCancelHelp(window=subdialog, helpSubject=NULL) # macro
subsetBox(window=top, model=FALSE) # macro
trim.blanks(text)
TwoLevelFactors(names)
twoLevelFactorsP(n=1)
UpdateModelNumber(increment=1)
variableListBox(parentWindow, variableList=Variables(), bg="white",
               selectmode="single", export="FALSE", initialSelection=NULL,
               listHeight = 4, title)
Variables(names)

# the following function is exported for technical reasons,
# but is not meant to be called directly

commanderPosition()

```

### Arguments

ask	ask for confirmation.
ask.save	ask whether to save contents of script and output windows.
bg	background color.
bindReturn	if TRUE, the <i>Return</i> key is bound to the onOK function in the dialog.

<code>boxes</code>	vector of quoted names for check boxes, used to generate each box and its associated variable.
<code>buttons</code>	vector of quoted names for buttons in a set of related radio buttons.
<code>class</code>	quoted name of class.
<code>columnspan</code>	number of dialog-box columns to be spanned by frame.
<code>command</code>	a character string that evaluates to an R command.
<code>dataSet, dsname</code>	the quoted name of a data frame in memory.
<code>default</code>	default button: if not specified, "ok" for "okcancel", "yes" for "yesno", and "ok" for "ok"; or look for a default method.
<code>envir</code>	the environment to be searched; should generally be left at the default.
<code>export</code>	export selection?
<code>expr</code>	expression constituting the body of the macro; typically a compound expression.
<code>flushModel</code>	set (or reset) the active model to NULL? Should normally be TRUE when the active data set is changed; an exception is when variables are simply added to, deleted from, or modified in the data set set.
<code>focus</code>	Tk window to get the focus.
<code>frame</code>	frame or quoted name for frame depending upon the function.
<code>generic</code>	quoted name of generic function.
<code>groupsBox</code>	listbox object for selecting groups variable.
<code>hasLhs</code>	does the model formula have a left-hand side?
<code>helpSubject</code>	the quoted name of a help subject, to be called as <code>help(helpSubject)</code> when the dialog <i>Help</i> button is pressed.
<code>icon</code>	Message-box icon.
<code>increment</code>	increment to model number; -1 to set back after error.
<code>initialLabel</code>	label for groups button before a selection is made.
<code>initialSelection</code>	index of item initially selected, 0-base indexing.
<code>initialValue</code>	for a set of related radio buttons.
<code>initialValues</code>	for a set of related check boxes.
<code>label</code>	label prefix for groups button after a selection is made.
<code>labels</code>	a vector of character strings to label a set of radio buttons or check boxes.
<code>listHeight</code>	Maximum number of elements displayed simultaneously in list box.
<code>loaded</code>	if TRUE, plug-in packages that are loaded are included in the vector of names returned.
<code>log</code>	echo command to the log window, as well as executing it and printing its output.
<code>message</code>	error (or other) message.
<code>mode</code>	mode of object to retrieve.

model	the name of a model, as a character string, or TRUE or FALSE, depending upon the function.
name	quoted name.
names	optional names to be stored.
n	number of items to check for.
object	an object (depends on context).
offset	in pixels, from top-left of Commander window.
onOK	function to execute when the <i>OK</i> button is pressed.
plotLinesByGroup	include a check box for plotting lines by group?
plotLinesByGroupsText	the label for the plot-lines-by-group check box.
positionLegend	include a check box for a legend?
preventGrabFocus	prevent the dialog box from grabbing the focus.
preventDoubleClick	prevent double-clicking from pressing the OK button, even when the double.click option is set; necessary for statistical modelling dialogs, which use double-clicking to build the model formula.
preventCrisp	prevent call to <code>tclServiceMode</code> , which (rarely) causes problems with some dialogs.
recall	function to call after error — usually the function that initiates the dialog.
release	release the focus if the <code>grab.focus</code> option has been set.
reportError	if TRUE, report an error message.
rows, columns	numbers of rows and columns of widgets in the dialog box.
values	vector of quoted values associated with radio buttons or check boxes.
selectmode	"single" or "multiple".
strict	if TRUE, only use first element of class vector.
text	a text string.
title	Window or dialog-box-element title.
type	quoted type of object to check; used to generate check-replace dialog box; or type of message to print in Message window.
value	an object to be stored.
variableList	a vector of variable names.
window, parentWindow	a Tk window.
x	an R object name, as a character string.
...	For <code>gettextRcmdr</code> , text string or vector of text strings to translate; for <code>defmacro</code> , arguments for the macro; otherwise disregard.

## Details

There are several groups of functions exported by the Rcmdr package and documented briefly here. To see how these functions work, it is simplest to examine the dialog-generating functions in the Rcmdr package.

*Executing and logging commands:* The functions `doItAndPrint`, `justDoIt`, and `logger` control the execution, logging, and printing of commands generated by menus and dialogs. `logger(command)` adds `command` to the log/script window and to the output window. `justDoIt(command)` causes `command` to be executed. `doItAndPrint(command)` does both of these operations, and also prints the output produced by the command.

*Checking for errors:* The function `isValidName` checks whether a character string specifies a valid name for an R object. The functions `checkActiveDataSet`, `checkActiveModel`, `checkFactors`, `checkNumeric`, `checkTwoLevelFactors`, and `checkVariables` check for the existence of objects and write an error message to the log if they are absent (or insufficiently numerous, in the case of different kinds of variables). The function `checkReplace` opens a dialog to query whether an existing object should be replaced. The function `checkMethod` checks whether a method exists for a particular generic that is appropriate for a particular object. The function `checkClass` checks whether an object is of a specific class. Both of these functions write error messages to the log if the condition fails. The function `errorCondition` reports an error to the user and (optionally) re-starts a dialog.

*Information:* Several functions return vectors of object names: `listAllModels`, `listDataSets`, `listGeneralizedLinearModels`, `listFactors`, `listLinearModels`, `listMultinomialLogitModels`, `listNumeric`, `listProportionalOddsModels`, `listTwoLevelFactors`, `listVariables`. The functions `activeDataSet` and `activeModel` respectively report or set the active data set and model. The function `packageAvailable` reports whether the named package is available to be loaded (or has possibly already been loaded). The function `exists.method` checks whether a method exists for a particular generic that is appropriate for a particular object, and returns TRUE or FALSE.

*Building dialog boxes:* Several functions simplify the process of constructing Tk dialogs: initializing a dialog box, `initializeDialog`, and completing the definition of a dialog box, `dialogSuffix`; a set of check boxes, `checkboxes`; a set of radio buttons, `radioButtons`; a list box with associated scrollbars and state variable, `variableListBox` (and the associated functions `getFrame` and `getSelection`); a button and subdialog for selecting a "grouping" variable, `groupsBox`; displaying the currently defined groups in a dialog, `groupsLabel`; a dialog-box structure for entering a model formula, `modelFormula`; a text box for entering a subsetting expression, `subsetBox`; *OK*, *Cancel*, and *Help* buttons for dialogs, `OKCancelHelp`, and subdialogs, `subOKCancelHelp`.

*Translating text:* The `getTextRcmdr` function simply passes its argument(s) to `gettext`, adding the argument `domain="R-Rcmdr"`.

*Miscellaneous:* The function `trim.blanks` removes spaces from the beginning and end of a character string. The function `installPlugin` installs an Rcmdr plug-in from a ZIP file or directory; this function may be used to create self-installing plug-ins in the form of packages.

Some of these functions, marked # `macro` under *Usage*, are "macro-like" in their behaviour, in that they execute in the environment from which they are called. These were defined with an adaptation (used with permission) of Thomas Lumley's `defmacro` function, described in Lumley (2001).

**Author(s)**

John Fox (jfox@mcmaster.ca)

**References**

T. Lumley (2001) Programmer's niche: Macros in R. *R News*, **1(3)**, 11–13.

---

RcmdrPager

*Pager for Text Files*

---

**Description**

This is a slightly modified version of the tkpager, changed to use the Rcmdr monospaced font and a white background.

**Usage**

```
RcmdrPager(file, header, title, delete.file)
```

**Arguments**

file	character vector of file(s) to be displayed.
header	for the beginning of each file.
title	for window
delete.file	delete file(s) on close.

**See Also**

[tkpager](#)

---

Recode

*Rcmdr Recode Dialog*

---

**Description**

The recode dialog is normally used to recode numeric variables and factors into factors, for example by combining values of numeric variables or levels of factors. It may also be used to produce new numeric variables. The Rcmdr recode dialog is based on the [recode](#) function in the `car` package.

### Details

The name of each new variable must be a valid R object name (consisting only of upper and lower-case letters, numerals, and periods, and not starting with a numeral).

Enter recode directives in the box near the bottom of the dialog. Directives are normally entered one per line, but may also be separated by semicolons. Each directive is of the form `input = output` (see the examples below). If an input value satisfies more than one specification, then the first (from top to bottom, and left to right) applies. If no specification is satisfied, then the input value is carried over to the result. NA is allowed on input and output. Factor levels are enclosed in double-quotes on both input and output.

Several recode specifications are supported:

**a single value** For example, `"missing" = NA`.

**several values separated by commas** For example, `7,8,9 = "high"`.

**a range of values indicated by a colon** For example, `7:9 = "high"`. The special values `low` and `high` may appear in a range. For example, `low:10=1`. Note that these values are unquoted.

**the special value `else`** everything that does not fit a previous specification. For example, `else=NA`. Note that `else` matches *all* otherwise unspecified values on input, including NA.

If all of the output values are numeric, and the "Make new variable a factor" check box is unchecked, then a numeric result is returned.

If several variables are selected for recoding, then each is recoded using the same recode directives. In this case, the name entered in the box labelled "New variable name or prefix for multiple recodes" will be prefixed to the name of each variable being recoded. Setting an empty prefix (i.e., "") will cause the recoded variables to replace the original variables.

### Author(s)

John Fox (jfox@mcmaster.ca)

### See Also

[recode](#)

---

reliability

*Reliability of a Composite Scale*

---

### Description

Calculates Cronbach's alpha and standardized alpha (lower bounds on reliability) for a composite (summated-rating) scale. Standardized alpha is for the sum of the standardized items. In addition, the function calculates alpha and standardized alpha for the scale with each item deleted in turn, and computes the correlation between each item and the sum of the other items.

**Usage**

```
reliability(S)

## S3 method for class 'reliability':
print(x, digits=4, ...)
```

**Arguments**

S	the covariance matrix of the items; normally, there should be at least 3 items and certainly no fewer than 2.
x	reliability object to be printed.
digits	number of decimal places.
...	not used: for compatibility with the print generic."

**Value**

an object of class reliability, which normally would be printed.

**Author(s)**

John Fox (jfox@mcmaster.ca)

**References**

N. Cliff (1986) Psychological testing theory. Pp. 343–349 in S. Kotz and N. Johnson, eds., *Encyclopedia of Statistical Sciences*, Vol. 7. Wiley.

**See Also**

[cov](#)

**Examples**

```
data(DavisThin)
reliability(cov(DavisThin))
```

**Description**

The **Rokkan** data frame has 16 rows and 5 columns.

Abridged from Ragin (1987):

The data was used by Rokkan (1970) in his work on nation building in Western Europe. Rokkan used a "configurational" approach that bears many similarities to the Boolean approach presented in this work. His main substantive interest was the growth of mass democracy and the emergence of different cleavage structures in Western European polities. One outcome that interested him was the division of some working-class movements in these countries following the Russian Revolution into internationally oriented wings and some into nationally oriented wings. He considered the distribution of this outcome important because of its implication for the future of working-class mobilization (and cleavage structures in general) in Western Europe

**Usage**

```
data(Rokkan)
```

**Format**

The dataset contains the following columns:

- C - National church (vs. state allied to Roman Catholic church)
- R - Significant Roman Catholic population and Roman Catholic participation in mass education
- L - State protection of landed interests
- E - Early state
- S - Major split in working-class movement provoked by Russian Revolution (outcome variable) (yes/no)

**Source**

Ragin, Charles C. 1987 *The Comparative Method. Moving beyond qualitative and quantitative strategies*, Berkeley: University of California Press, pp.129

**References**

Rokkan, Stein 1970 *Citizens, Elections, Parties*, New York: McKay

---

stem.leaf

*Stem-and-Leaf Display*

---

**Description**

Creates a classical ("Tukey-style") stem-and-leaf display.

**Usage**

```
stem.leaf(data, unit, m, Min, Max,
          rule.line = c("Dixon", "Velleman", "Sturges"),
          style = c("Tukey", "bare"), trim.outliers = TRUE, depths = TRUE,
          reverse.negative.leaves = TRUE)

## S3 method for class 'stem.leaf':
print(x, ...)
```

**Arguments**

<code>data</code>	a numeric vector.
<code>unit</code>	leaf unit, as a power of 10 (e.g., 100, .01); omit to let the function choose the unit.
<code>m</code>	number of parts (1, 2, or 5) into which each stem should be divided; omit to let the function choose the number of parts/stem.
<code>Min</code>	smallest non-outlying value; omit for automatic choice.
<code>Max</code>	largest non-outlying value; omit for automatic choice.
<code>rule.line</code>	the rule to use for choosing the desired number of lines in the display; "Dixon" = $10 \cdot \log_{10}(n)$ ; "Velleman" = $2 \cdot \sqrt{n}$ ; "Sturges" = $1 + \log_2(n)$ ; the default is "Dixon".
<code>style</code>	"Tukey" (the default) for "Tukey-style" divided stems; "bare" for divided stems that simply repeat the stem digits.
<code>trim.outliers</code>	if TRUE (the default), outliers are placed on LO and HI stems.
<code>depths</code>	if TRUE (the default), print a column of "depths" to the left of the stems; the depth of the stem containing the median is the stem-count enclosed in parentheses.
<code>reverse.negative.leaves</code>	if TRUE (the default), reverse the leaves on negative stems (so, e.g., the leaf 9 comes before the leaf 8, etc.).
<code>x</code>	an object of class <code>stem.leaf</code> to be printed.
<code>...</code>	not used: for compatibility with the generic print function.

**Details**

Unlike the `stem` function in the `base` package, this function produces classic stem-and-leaf displays, as described in Tukey's *Exploratory Data Analysis*. Outliers are determined using the rule for boxplots (see `boxplot.stats`).

**Value**

Returns on object of class `stem.leaf`, which normally would be printed.

**Author(s)**

Peter Wolf, slightly modified by John Fox (jfox@mcmaster.ca) with the original author's permission.

**References**

Tukey, J. *Exploratory Data Analysis*. Addison-Wesley, 1977.

**See Also**

[stem](#)

**Examples**

```
data(Prestige)
stem.leaf(Prestige$income)
```

---

Stokke

*Causal mechanisms and regime effectiveness*

---

**Description**

The **Stokke** data frame has 9 rows and 6 columns.

The purpose of introducing mechanisms in regime research, is partly to allow detailed examination of the various ways in which regimes may affect behaviour: this could be coined a magnifying purpose. The second objective is the methodological one of facilitating systematic comparison of cases by constituting them in ways that make them sufficiently homogeneous to permit employment of available comparative techniques. The cases presented in this data stress on "shaming" as a causal mechanism for international resource management. Shaming highlights attempts to bring about a change in problem-related behaviour not by material rewards or punishment but by exposing certain practices to third parties whose opinion matters to the target of shaming

**Usage**

```
data(Stokke)
```

**Format**

The dataset contains the following columns:

A	-	Advice
C	-	Commitment
S	-	Shadow of the future
I	-	Inconvenience
R	-	Reverberation
Y	-	Success (outcome variable)

**Source**

<http://www.compass.org>

**References**

Stokke, Olav Schram 2004 *Boolean Analysis, Mechanisms, and the Study of Regime Effectiveness* in Underdal, Arild and Young, Oran R. (eds.) *Methodological Challenges and Research Strategies* Dordrecht: Kluwer Academic, pp. 87-119

---

Yamasaki

*The problem of contradictory simplifying assumptions*

---

**Description**

The **Yamasaki** data frame has 6 rows and 6 columns.

In Qualitative Comparative Analysis (QCA) of social data, the generation of parsimonious explanatory equations is enhanced by the inclusion of "logical configurations". Even if this procedure proves to be very useful, it also raises various methodological issues. Among them, the tricky problem of "contradictory simplifying assumptions" has remained largely unexplored. Yet the careful control of this obstacle is crucial for any QCA to be successful, not only because contradictory assumptions are inducing wrong conclusions, but also because their resolution can generate most interesting results. Hence, our contribution aims at enlightening this difficulty, as well as designing an efficient way to overcome it. In this perspective, we start from data collected for a comparative research on "the political feasibility of an unconditional basic income" in six OECD countries (1980-2002). After having briefly stated the core elements of the research question, six operational variables are defined (section 1). On this basis, we conduct a Boolean analysis and comment the various 'feasibility scenarios' generated by the QCA 3.0 software (section 2). Starting from these first results, we identify contradictory simplifying assumptions used by the software, and discuss possible solutions to this problem. New results are then generated (section 3). In the conclusion, we shortly discuss the general implications of this methodological problem

**Usage**

`data(Yamasaki)`

**Format**

The dataset contains the following columns:

POSTMAT	the level of postmodern values in a society takes the value "1" in the Dutch and Finnish cases, and "0" in the others
NONGHENT	the absence of a Ghent system is coded "1" and "0" when the Ghent system exists in the country
MOVEMENT	in the presence of a social movement advocating Basic Income, the variable will be coded "1"
UNITARY	based on the Lijphart index (1.0 to 5.0, from unitary to federal and decentralised states), the variable is coded "0" for cases in the three

SOCIAL following categories: semi-federal states (3.0), federal and centralized states (4.0), and federal and decentralized states (5.0), and "1" otherwise. based on Esping-Andersen's categorisation of welfare regimes, the variable is coded "1" for non-liberal countries and "0" otherwise

AGENDA the outcome variable, coded "1" when Basic Income has been considered as a serious alternative at the governmental level

**Source**

<http://www.compass.org>

**References**

Vanderborght, Yannick and Yamasaki, Sakura 2003 *The Problem of Contradictory Simplifying Assumptions in Qualitative Comparative Analysis (QCA)*, Paper presented at the ECPR General conference, 18-21 Sept., Marburg, Germany

Vanderborght, Yannick and Yamasaki, Sakura 2004 *Des cas logiques...contradictaires? Un piège de l'AQQC dejoué à travers l'étude de la faisabilité politique de l'Allocation Universelle*, *Revue Internationale de Politique Comparée*, Vol.11, pp.51-66

# Index

## \*Topic **datasets**

- big10, 5
- dozen12, 12
- Ebbinghaus, 12
- even11, 13
- lucky13, 18
- MI.15, 18
- mighty14, 18
- Osa, 20
- Rokkan, 32
- Stokke, 35
- Yamasaki, 36

## \*Topic **hplot**

- Hist, 15
- plotMeans, 21

## \*Topic **manip**

- bin.var, 5
- Compute, 11
- Recode, 30

## \*Topic **misc**

- assignCluster, 4
- colPercents, 6
- Commander, 6
- hierarchicalCluster, 14
- KMeans, 16
- numSummary, 19
- partial.cor, 21
- Plugins, 23
- QCAGUI-package, 2
- Rcmdr.sciviews-specific, 23
- Rcmdr.Utilities, 24
- RcmdrPager, 30
- reliability, 31
- stem.leaf, 33

## \*Topic **models**

- generalizedLinearModel, 13
- linearModel, 17

## \*Topic **package**

- Rcmdr-package, 2

- activateMenus (*Rcmdr.Utilities*), 24
- ActiveDataSet (*Rcmdr.Utilities*), 24
- activeDataSet (*Rcmdr.Utilities*), 24
- activeDataSetEdit (*Rcmdr.sciviews-specific*), 23
- activeDataSetP (*Rcmdr.Utilities*), 24
- activeDataSetView (*Rcmdr.sciviews-specific*), 23
- ActiveModel (*Rcmdr.Utilities*), 24
- activeModel (*Rcmdr.Utilities*), 24
- activeModelP (*Rcmdr.Utilities*), 24
- Arithmetic, 12
- assignCluster, 4
  
- big10, 5
- bin.var, 5
- boxplot.stats, 34
  
- checkActiveDataSet (*Rcmdr.Utilities*), 24
- checkActiveModel (*Rcmdr.Utilities*), 24
- checkBoxes (*Rcmdr.Utilities*), 24
- checkClass (*Rcmdr.Utilities*), 24
- checkFactors (*Rcmdr.Utilities*), 24
- checkMethod (*Rcmdr.Utilities*), 24
- checkNumeric (*Rcmdr.Utilities*), 24
- checkReplace (*Rcmdr.Utilities*), 24
- checkTwoLevelFactors (*Rcmdr.Utilities*), 24
- checkVariables (*Rcmdr.Utilities*), 24
- closeCommander (*Rcmdr.Utilities*), 24

- closeDialog (*Rcmdr.Utilities*), 24
- colPercents, 6
- Commander, 6, 23
- commanderPosition  
(*Rcmdr.Utilities*), 24
- CommanderWindow  
(*Rcmdr.Utilities*), 24
- Comparison, 14, 17
- Compute, 11
- cor, 21
- cov, 32
- cut, 5
- cutree, 4
  
- dataSetsP (*Rcmdr.Utilities*), 24
- defmacro (*Rcmdr.Utilities*), 24
- dialogSuffix (*Rcmdr.Utilities*), 24
- dist, 14, 15
- doItAndPrint (*Rcmdr.Utilities*), 24
- dozen12, 12
  
- Ebbinghaus, 12
- errorCondition (*Rcmdr.Utilities*),  
24
- even11, 13
- exists.method (*Rcmdr.Utilities*),  
24
  
- Factors (*Rcmdr.Utilities*), 24
- factorsP (*Rcmdr.Utilities*), 24
- family, 14
  
- generalizedLinearModel, 13
- getFrame (*Rcmdr.Utilities*), 24
- getRcmdr (*Rcmdr.Utilities*), 24
- getSelection (*Rcmdr.Utilities*), 24
- gettext, 29
- gettextRcmdr (*Rcmdr.Utilities*), 24
- glm, 13, 14
- glmP (*Rcmdr.Utilities*), 24
- GrabFocus (*Rcmdr.Utilities*), 24
- groupsBox (*Rcmdr.Utilities*), 24
- groupsLabel (*Rcmdr.Utilities*), 24
  
- hclust, 4, 14, 15
- hclustSolutionsP  
(*Rcmdr.Utilities*), 24
- hierarchicalCluster, 14
- Hist, 15
  
- hist, 15, 16
  
- initializedDialog  
(*Rcmdr.Utilities*), 24
- interaction.plot, 22
- is.SciViews  
(*Rcmdr.sciviews-specific*),  
23
- is.valid.name (*Rcmdr.Utilities*),  
24
  
- justDoIt (*Rcmdr.Utilities*), 24
  
- KMeans, 4, 16
- kmeans, 4, 5, 17
  
- linearModel, 17
- listAllModels (*Rcmdr.Utilities*),  
24
- listDataSets (*Rcmdr.Utilities*), 24
- listFactors (*Rcmdr.Utilities*), 24
- listGeneralizedLinearModels  
(*Rcmdr.Utilities*), 24
- listLinearModels  
(*Rcmdr.Utilities*), 24
- listMultinomialLogitModels  
(*Rcmdr.Utilities*), 24
- listNumeric (*Rcmdr.Utilities*), 24
- listPlugins (*Rcmdr.Utilities*), 24
- listProportionalOddsModels  
(*Rcmdr.Utilities*), 24
- listTwoLevelFactors  
(*Rcmdr.Utilities*), 24
- listVariables (*Rcmdr.Utilities*),  
24
  
- lm, 17
- lmP (*Rcmdr.Utilities*), 24
- logger (*Rcmdr.Utilities*), 24
- LogWindow (*Rcmdr.Utilities*), 24
- lucky13, 18
  
- mean, 19
- Message (*Rcmdr.Utilities*), 24
- MessagesWindow (*Rcmdr.Utilities*),  
24
- MI.15, 18
- mighty14, 18
- modelFormula (*Rcmdr.Utilities*), 24
- modelsP (*Rcmdr.Utilities*), 24

- Numeric (*Rcmdr.Utilities*), 24
- numericP (*Rcmdr.Utilities*), 24
- numSummary, 19
- OKCancelHelp (*Rcmdr.Utilities*), 24
- optionAttachDataSet
  - (*Rcmdr.sciviews-specific*), 23
- optionLogCommand
  - (*Rcmdr.sciviews-specific*), 23
- optionSortVariables
  - (*Rcmdr.sciviews-specific*), 23
- Osa, 20
- OutputWindow (*Rcmdr.Utilities*), 24
- packageAvailable
  - (*Rcmdr.Utilities*), 24
- partial.cor, 21
- plotMeans, 21
- Plugins, 10, 11, 23
- print.numSummary (*numSummary*), 19
- print.reliability (*reliability*), 31
- print.stem.leaf (*stem.leaf*), 33
- putRcmdr (*Rcmdr.Utilities*), 24
- QCAGUI (*QCAGUI-package*), 2
- QCAGUI-package, 2
- quantile, 19
- radioButtons (*Rcmdr.Utilities*), 24
- Rcmdr (*Rcmdr-package*), 2
- Rcmdr-package, 2
- Rcmdr.sciviews-specific, 11
- Rcmdr.sciviews-specific, 23
- Rcmdr.Utilities, 8, 24
- RcmdrPager, 30
- RcmdrTclSet (*Rcmdr.Utilities*), 24
- RcmdrTkmessageBox
  - (*Rcmdr.Utilities*), 24
- Recode, 30
- recode, 30, 31
- refreshStatus
  - (*Rcmdr.sciviews-specific*), 23
- reliability, 31
- rglLoaded (*Rcmdr.Utilities*), 24
- Rokkan, 32
- rowPercents (*colPercents*), 6
- sd, 19
- stem, 35
- stem.leaf, 33
- Stokke, 35
- subOKCancelHelp
  - (*Rcmdr.Utilities*), 24
- subsetBox (*Rcmdr.Utilities*), 24
- svCommander
  - (*Rcmdr.sciviews-specific*), 23
- svlogger
  - (*Rcmdr.sciviews-specific*), 23
- tkfocus
  - (*Rcmdr.sciviews-specific*), 23
- tkpager, 30
- totPercents (*colPercents*), 6
- trim.blanks (*Rcmdr.Utilities*), 24
- TwoLevelFactors
  - (*Rcmdr.Utilities*), 24
- twoLevelFactorsP
  - (*Rcmdr.Utilities*), 24
- UpdateModelNumber
  - (*Rcmdr.Utilities*), 24
- variableListBox
  - (*Rcmdr.Utilities*), 24
- Variables (*Rcmdr.Utilities*), 24
- Yamasaki, 36