Package 'Rcmdr'

April 11, 2026

Version 2.12.2

Date 2026-04-12

Title R Commander

Encoding UTF-8

Depends R (>= 3.5.0), grDevices, graphics, methods, stats, utils, splines, RcmdrMisc (>= 2.10.1), car (>= 3.1-0), effects (>= 4.0-3)

Imports tcltk, tcltk2 (>= 1.2-6), abind, relimp (>= 1.0-5), lme4, tools

Suggests aplpack, boot, colorspace, e1071, foreign, grid, Hmisc, knitr, lattice, leaps, lmtest, markdown, MASS, mgcv, multcomp (>= 0.991-2), nlme, nnet, nortest, readxl, rgl (>= 0.110.2), rmarkdown (>= 0.9.5), sem (>= 2.1-1)

ByteCompile yes

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

License GPL (>= 2)

URL https://github.com/RCmdr-Project/rcmdr, https://www.r-project.org, https://www.john-fox.ca/RCommander/index.html

RoxygenNote 7.3.3

NeedsCompilation no

Author John Fox [aut],
Milan Bouchet-Valat [aut],
Manuel Munoz-Marquez [aut, cre],
Liviu Andronic [ctb],
Michael Ash [ctb],
Theophilius Boye [ctb],
Stefano Calza [ctb],
Andy Chang [ctb],
Vilmantas Gegzna [ctb],
Philippe Grosjean [ctb],

Richard Heiberger [ctb],
Yoshinobu Kanda [ctb],
Kosar Karimi Pour [ctb],
G. Jay Kerns [ctb],
Renaud Lancelot [ctb],
Matthieu Lesnoff [ctb],
Uwe Ligges [ctb],
Samir Messad [ctb],
Martin Maechler [ctb],
Robert Muenchen [ctb],
Duncan Murdoch [ctb],
Erich Neuwirth [ctb],
Dan Putler [ctb],
Brian Ripley [ctb],
Miroslav Ristic [ctb],
Peter Wolf [ctb],
Kevin Wright [ctb]

Maintainer Manuel Munoz-Marquez <manuel.munoz@uca.es>

Repository CRAN

Date/Publication 2026-04-11 06:21:29 UTC

Contents

Rcmdr-package *R Commander*

Description

R Commander

Details

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

Package:	Rcmdr
Version:	2.12.2
Date:	2026-04-12
Depends:	R (>= 3.5.0), grDevices, graphics, methods, stats, utils, splines, RcmdrMisc (>= 2.10.1), car (>= 3.1-0), effec
Imports:	tcltk, tcltk2 (>= 1.2-6), abind, relimp (>= 1.0-5), lme4, tools
Suggests:	aplpack, boot, colorspace, e1071, foreign, grid, Hmisc, knitr, lattice, leaps, lmtest, markdown, MASS, mgcv,
ByteCompile:	yes
License:	GPL (>= 3)
URL:	https://github.com/RCmdr-Project/rcmdr, https://www.r-project.org, https://www.john-fox.ca/RCommander/i

Translations

The R Commander comes with translations from English into several other languages. I am grateful to the following individuals and groups for preparing these translations: Basque, Jose Ramon Rueda; Brazilian Portuguese, Adriano Azevedo-Filho and Marilia Sa Carvalho; Catalan, Manel Salamero; Chinese, Tsungwu Ho, Frank C. S. Liu, and Cheng-shun Lee; Chinese (Simplified), Shulin Yang; French, Philippe Grosjean and Milan Bouchet-Valat; Galician, Anton Meixome; German: Friedrich Leisch and Gerhard Schoen; Greek: Vasileios Dimitropoulos, Anastasios Vikatos, and Andreas Vikatos; Indonesian, I Made Tirta; Italian, Stefano Calza; Japanese, Takaharu Araki; Korean, Chel Hee Lee, Dae-Heung Jang, and Shin Jong-Hwa; Polish, Lukasz Daniel; Romanian, Adrian Dusa; Russian, Alexey Shipunov; Slovenian, Jaro Lajovic and Matjaz Jeran; Spanish, Spanish R-UCA Project, http://knuth.uca.es/R.

Author(s)

John Fox, Manuel Munoz-Marquez, and Milan Bouchet-Valat, with contributions from Liviu Andronic, Michael Ash, Theophilius Boye, Stefano Calza, Andy Chang, Vilmantas Gegzna, Philippe Grosjean, Richard Heiberger, G. Jay Kerns, Renaud Lancelot, Matthieu Lesnoff, Uwe Ligges, Samir Messad, Martin Maechler, Robert Muenchen, Duncan Murdoch, Erich Neuwirth, Dan Putler, Brian Ripley, Miroslav Ristic, Peter Wolf, and Kevin Wright.

Maintainer: Manuel Munoz-Marquez <manuel.munoz@uca.es>

References

Fox, J. (2017) *Using the R Commander: A Point-and-Click Interface for R*. Chapman and Hall/CRC Press.

Fox, J. (2005) The R Commander: A Basic Statistics Graphical User Interface to R. *Journal of Statistical Software*, **14(9)**: 1–42.

`AuxiliarySoftware` *Install R Commander Auxiliary Software*

Description

Install R Commander Auxiliary Software

Usage

```
installSoftware()
```

Details

Install Pandoc and LaTeX to increase the capabilities of the R Commander.

The capabilities of the R Commander can be enhanced by installing additional software. The R Commander *will work* without this software but some features will not be activated. The following addiitional software can conveniently be installed via the R Commander *Tools* > *Install auxiliary software* menu. The resulting dialog box will take you to websites where the software can be downloaded. This menu item will only be displayed if one or more of these software packages are missing.

Here are the details:

- *Pandoc*: The Pandoc documentation-conversion software is used by the R Commander to generate HTML (web), PDF, and Word files from the editable R Markdown document that is created by default during an R Commander interactive session. Pandoc is required by the **rmarkdown** package, which, along with the **knitr** package, performs these conversions. In the absence of Pandoc, R Markdown documents in the R Commander are processed by the older **markdown** package, which is capable only of producing HTML output. Pandoc is available from https://pandoc.org/installing.html.

  On Windows systems, Pandoc installs into a non-standard location in your user directory, typically `C:\Users\<your user id>\AppData\Local\Pandoc\`, and then places this subdirectory on your system path. You may have to reboot for the change to your path to take effect, and I have found it necessary on two Windows 10 systems to re-run the Pandoc installer, first to uninstall Pandoc, and then to re-install it before it would be work.
- *LaTeX*: The LaTeX technical-typesetting system is required by the R Commander for PDF output from R Markdown or knitr documents produced during interactive R Commander sessions. In the absence of LaTeX, direct PDF output is unavailable. Complete LaTeX systems are available for the various platforms that support R and the R Commander, including MikTeX from https://miktex.org/download for Windows systems; MacTeX from https://www.tug.org/mactex/ for MacOSX; and various sources (see https://www.latex-project.org/get/) for Linux/Unix systems.

See Also

Commander

CFA *Rcmdr Confirmatory Factor Analysis Dialog*

Description

Rcmdr Confirmatory Factor Analysis Dialog

Usage

```
CFA()
```

Details

The CFA dialog is used to create and fit a confirmatory factor analysis model via the `cfa` and `sem` functions in the **sem** package.

Select two or more variables for each factor by *Control*-clicking on their names in the variable-list box. Optionally give the factor a name; this must be a valid R name. Then press the *Define factor* button.

Continue in this manner until all factors are specified. Note that if there are not at least two unique variables selected for each factor, the model will probably be underidentified, causing `sem` to fail.

The radio buttons at the top of the dialog may be used to analyze either the correlation matrix or covariance matrix of the observed variables; to specify either correlated or orthogonal factors; and to identify the model either by setting the factor variance to 1 or by setting the first loading for each factor to 1 (establishing a "reference indicator" for the factor). A check box is provided to compute robust standard errors and tests.

Author(s)

John Fox

See Also

`sem`, `cfa`

`Commander` *R Commander*

Description

Start the R Commander GUI (graphical user interface)

Usage

```
Commander()
```

Details

Getting Started

For more detailed information about getting started, see *Help -> Introduction to the R Commander* from the R Commander menus or Fox (2017).

The default R Commander interface consists of (from top to bottom) a menu bar, a toolbar, a code window with script and R Markdown tabs, 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 tab. 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, SAS XPORT data set; or an Excel spreadsheet. 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 tab (the initially empty text window immediately below the toolbar), and commands and output appear in the Output window (the initially empty text window below the Script tab). Commands that don't require direct user interaction (such as interactive identification of points on a graph) are also used to create an R Markdown document in the tab of the same name. When the R Markdown tab is in front, pressing the "Generate HTML report" button compiles the document to create an html page with input and output, which opens in a web browser. To alter these and other defaults, see the information below on configuration. Note, for example, that the **knitr** package can be used to create a LaTeX document to be compiled to a PDF report, as an alternative to — or in addition to — an R Markdown document (see the `use.knitr` option below).

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 in turned on, R commands that are generated from the menus and dialog boxes are entered into the Script and R Markdown tabs in the Commander. You can edit these commands in the normal manner and can also type new commands. You can also type explanatory text in the R Markdown tab. Individual commands in the Script tab can be continued over more than one line, but 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 and R Markdown tabs can be saved during or at the end of the session, and a saved script or R Markdown document can be loaded into the respective tabs. The contents of the Output window can also be edited or saved to a text file. Finally, editing operations also work in the Messages window.

To re-execute a command or set of commands in the Script tab, 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", or *Control-Tab*). If no text is selected, the *Submit* button (or *Control-R* or *Control-Tab*) 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 tab, R Markdown tab, knitr tab, Output window, or Messages window. Edit functions such as search are performed in the Script tab unless you first click in another tab or window to make it active.

Pressing *Control-S* will save the Script tab, R Markdown tab, knitr tab, or Output window.

Pressing *Control-A* selects all of the text in the Script tab, R Markdown tab, knitr tab, Output window, or Messages window.

In addition, the following Control-key combinations work in these tabs and windows: *Control-X*, cut; *Control-C*, copy; *Control-V*, insert; *Control-Z* or *Alt-Backspace*, undo; and *Control-W*, redo.

Under Mac OS X, the *command* key may be used in place of the *Control* key, though the latter works as well.

Right-clicking the mouse (clicking button 3 on a three-button mouse, or *Control*-left-clicking) in the tabs or windows brings up a "context" menu with the *Edit*-menu items, plus (in the Script, R Markdown, and knitr tabs) a *Submit* item.

You can open a larger editor window with the document in the Markdown or knitr tab by making the corresponding selection from the *Edit* menu, the right-click context menu when the cursor is in the tab, or by pressing *Control-E* when the cursor is in the tab.

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.

You can cancel an R Commander dialog box by pressing the *Esc* key.

Most R Commander dialogs remember their state when this is appropriate, and can be restored to pristine state by pressing the Reset button.

Some R Commander dialogs have an Apply button that will execute the command generated by the dialog and then re-open the dialog in its previous state.

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

Customization and Configuration

The preferred way of customizing the R Commander is to write a plug-in package: see `help("Plugins")`.

Alternatively, 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`.

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, submenus, or menu-items. Alternatively, you can edit the source package and recompile it.

To reiterate, however, the preferred procedure is to write an R Commander plug-in package.

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:

`ask.to.exit` if `TRUE` (the default), then the user is asked whether he or she wants to exit the **Rcmdr**; if this option is set to `FALSE`, then the subsequent option is also set to `FALSE`.

`ask.on.exit` if `TRUE` (the default), then the user is asked whether to save the script file, R Markdown file, and output file when the **Rcmdr** exits.

`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`, unless the R Commander is running under RStudio, in which case the default is `TRUE`.

`crisp.dialogs` If `TRUE`, dialogs should appear on the screen fully drawn, rather than built up widget by widget. Prior to R 2.6.1, this option only works on the Windows version of R, but should in any event be harmless. The default is `TRUE`. If you encounter stability problems, try setting this option to `FALSE`.

`default.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.

`default.font.family` The default font for GUI elements such as menus and text labels, in the form of a Tk font family specification, given in a character string. For example, `"Helvetica"` specifies the sans-serif Helvetica font family. The default is taken from the `TkDefaultFont`. Normally a sans-serif font should be used.

`default.font.size` The size, in points, of the default font. The default is 10 on non-Windows system and the size of the system font on Windows. To set the font size for R input and output, see the `log.font.size` option. The **Rcmdr** `scale.factor` option may also be used to control font size.

`discreteness.theshold` should be a positive integer; if greater than `0` (which is the default), the maximum number of distinct values for a numeric variable to be considered discrete; if `0` (or smaller), the threshold is taken as the smallest of 100, twice the squareroot of the number of cases in the active data set (n), and 10 times log10(n).

`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`.

`editDataset.threshold` If the number of values in the current data set exceed this value (the default is 10000), then the standard R data editor is used in preference to the R Commander `editDataset` editor.

`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`.

`help_type` This Rcmdr option takes precedence over the global R `help_type` option (see `options` and `help`), and by default is set to `"html"`.

`iconify.commander` If `TRUE`, the *Commander* window is minimized on startup; the default is `FALSE`.

`length.output.stack` The R Commander maintains a list of output objects, by default including the last several outputs; the default length of the output stack is 10. `popOutput()` "pops" (i.e., returns and removes) the first entry of the output stack. Note that, as a stack, the queue is LIFO ("last in, first out").

`length.command.stack` The R Commander also maintains a list of commands that is managed similarly; the default length of this stack is also 10.

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

`log.font.family` The font family to be used for text in the script window, output window, messages window, etc., specified as a character vector giving a Tk font family. This should normally be a monospaced font like `"Courier"`. The default is taken from the `TkFixedFont`.

`log.font.size` The font size, in points, to be used in the script window, in the output window, messages window, in recode dialogs, and in compute expressions — that is, where a monospaced font is used. The default is 10. Alternatively the **Rcmdr** `scale factor` option may also be used to control font size.

`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.

`messages.height` The height of the messages window, in lines. The default is 4.

`model.case.deletion` if `TRUE` (the default is `FALSE`), include a text box for case deletion in statistical-model dialog boxes (e.g., for linear models).

`minimum.width` The minimum width, in pixels, for the main R Commander windows; the default is `1000`.

`minimum.height` The minimum height, in pixels, for the main R Commander windows; the default is `400`.

`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.

`number.messages` If `TRUE`, the default, messages in the messages window are numbered.

`open.graphics.devices` If `TRUE` (the default is `FALSE`), open the system graphics device and (if 3D RGL graphics are used) the RGL graphics device when the R Commander starts.

`open.markdown.editor` If `TRUE` (the default is `FALSE`), open the R Markdown editor when the R Commander starts.

`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 `""`, which lets the Tk window manager decide where to place the window; for example, `"+20+20"` should put the window near the upper-left corner of the screen, `"-20+20"` near the upper-right corner, though this doesn't appear to work reliably on Windows systems.

`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.

`prefixes` A four-item character vector to specify the prefixes used when output is directed to the R console; the default is `c("Rcmdr> ", "Rcmdr+ ", "RcmdrMsg: ", "RcmdrMsg+ ")`.

`quit.R.on.close` if `TRUE`, both the Commander and R are exited when the Commander window is closed. The default is `FALSE`, in which case only the Commander is exited (and can be restarted by the command `Commander()`).

`RcmdrEnv.on.path` If `TRUE` (the default is `FALSE`), the environment in which R Commander state information is stored is placed on the search path. Some plug-ins, at least until they are updated, may require this setting.

`retain.messages` If `TRUE` (the default), 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".

`retain.selections` If `TRUE` (the default), dialogs remember their previous state, where appropriate, as long as the data set isn't changed; some dialogs, e.g., for probabilities, retain selections even when the data set chanages.

`RExcelSupport` If `TRUE` (the default is `FALSE`), menus and output are handled by Excel.

`rmarkdown.output` Values of several options for converting R Markdown to a document file. The default for this option is `TRUE`, which corresponds to `markdown.output=list(command.sections=TRUE, section.level=3, toc=TRUE, toc_float=TRUE, toc_depth=3, number_sections=FALSE, translate.rmd.headers=TRUE)`. The sub-option `command.sections` controls whether most R commands produce sections in the R Markdown document; the sub-option `section.level` controls the level of the sections that are created; the sub-option `translate.rmd.headers` controls whether the headers are translated from English into another language, if a translation is available; and the other sub-options are standard for `rmarkdown`. The `toc_float`, `toc_depth`, and `number_sections` sub-options are only effective if Pandoc is installed.

`rmd.output.format` The output file type for R Markdown documents if pandoc is installed; one of `"html"` (the default), `"pdf"` (requires LaTeX), `"docx"` (Word), or `"rtf"` (rich text file).

`rmd.template` The quoted path to a `.Rmd` file to serve as a template for R code and output. The default is to use a template included with the package.

`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`.

`scientific.notation` Higher numbers cause ordinary (decimal) notation to be increasingly preferred to scientific notation for representing very small and very large numbers; correspond to the `scipen` option in R: see `options`. The default is `5`, while the standard default in R is `0` (where 0 means that scientific notation is used whenever the resulting printed representation of a number is smaller in scientific than in standard notation).

`showData.threshold` a vector with 2 entries, defaulting to `c(20000, 100)`. If the number of cases in the active data set exceeds the first number (default, 20,000) or the number of variables exceeds the second number (default, 100), then `View()` rather than `showData()` is used to display the data set. The reason for the option is that `showData()` is very slow when the number of cases or variables is large; setting the threshold to `c(0, 0)` suppresses the use of `showData` altogether. It's necessary to use `showData` however for the view of the active data set to be updated dynamically when, e.g., a variable is added.

`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 as been solved).

`sort.names` Set to `TRUE` (the default) if you want variable names to be sorted alphabetically in variable lists.

`suppress.icon.images` Set to `TRUE` to suppress the icon images in dialog OK, Cancel, Reset, and Help buttons; the default is `FALSE`.

`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 and Mac OS X 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) 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.

`theme` A ttk theme to control the overall style of the Commander GUI; should be one of the themes returned by `tcltk2::tk2theme.list()`. The default theme varies by operating system, and can be discovered by entering the command `tcltk2::tk2theme()` in a fresh R session.

`title.color` Color for the titles of some widgets, such as variable-list boxes; can be given as a color name, such as `"blue"` or as an RGB value, such as `"#0000FF"`. The default is the standard color for ttk label frames, unless that is `"#000000"` or `"black"`, in which case `"blue"` is used instead.

`tkwait.commander` 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.commander` 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.

`tkwait.dialog` If `TRUE` (the default is `FALSE`), R will wait until an R Commander dialog is closed. This has the disadvantage of preventing help pages from being displayed until a dialog is closed in the Mac OS X R.app and in RStudio. This was also the standard behavior of the R Commander in earlier versions and is provided for compatibility with previous behavior. If this option is `TRUE`, then the R Commander data editor is disabled in favor of the standard R platform-specific data editor, and the new-data-set menu item is suppressed.

`use.knitr` If `TRUE` (the default is `FALSE`), a knitr `.Rnw` LaTeX document is created in a tab of the main Commander window; this document can be compiled into `.tex` and `.pdf` reports via the `knit2pdf` function in the **knitr** package.

`use.markdown` If `TRUE` (the default is the negation of the `use.knitr` argument), an R Markdown document is created, which can be compiled into an HTML, PDF, Word, or rich text file report.

`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.

`"valid.classes"` The classes of variables that the R Commander recognizes, in addition to numeric data; other variables in a data set will be suppressed. The default is `"factor"`, `"ordered"`, `"character"`, `"logical"`, `"POSIXct"`, `"POSIXlt"`, `"Date"`, `"chron"`, `"yearmon"`, `"yearqtr"`, `"zoo"`, `"zooreg"`, `"timeDate"`, `"xts"`, `"its"`, `"ti"`, `"jul"`, `"timeSeries"`, `"fts"`, `"Period"`, `"hms"`, `"difftime"`).

`variable.list.height` the number of items (typically variables) to display in list boxes; longer lists may be viewed by scrolling. The default is 6.

`variable.list.width` a two-item vector controlling the width of list boxes, in characters, giving the minimum and maximum width to display; the default is `c(20, Inf)`. If the widest item name falls in this range, then its number of characters determines the width of the box. *Note:* This specification works only approximately.

`warning.text.color` Color for warning messages; the default is `"darkgreen"`.

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

If you want always to launch the R Commander when R starts up, you can include the following code in one of R's start-up files (e.g., in the `Rprofile.site` file in R's `etc` subdirectory):

```
local({
old <- getOption("defaultPackages")
options(defaultPackages = c(old, "Rcmdr"))
})
```

R Commander options can also be permanently set in the same manner. For more information about R initialization, see `?Startup`.

Warning

The R Commander Script window does not provide a true console to R, and may have 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. If you encounter any problems with the Script tab, however, I'd appreciate it if you brought them to my attention.

Platform-Specific Issues

Under Windows, the **Rcmdr** package can be run under the *Rgui* in the SDI (single-document interface) mode, or under `rterm.exe`. You might experience problems running the **Rcmdr** under `ESS` with NTEmacs or XEmacs, or under other R consoles. The R Commander can be run under the *Rgui* in MDI (multiple-document interface) mode but it is relatively inconvenient to do so and isn't recommended.

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. I have not seen this behavior in some time and the problem may have been solved.

Under Mac OS X Mavericks and later, the R Commander may appear to freeze or hesitate when run under *R.app* if the *R.app* window is hidden and "app nap" is turned on. It is recommended that app nap be turned off for *R.app*, which can be most conveniently done via the R Commander *Tools* menu. The app nap setting is permanent until changed and so the current setting will apply whether or not the R Commander is used. When R is first installed, app nap will be on for *R.app*. The **tcltk** package requires that X Windows is installed under Mac OS X, and as a consequence the **Rcmdr** package, which depends on **tcltk**, will not load if X Windows is absent. X Windows for Mac OS X may be obtained from https://www.xquartz.org/.

Note

On startup, the R Commander sets `options(na.action=na.exclude)`; this is done so that observation statistics such as residuals can be properly added to the active data set when there are missing values. The option is reset to its pre-existing value when the Commander exits. Some functions may not work properly when the default `na.action` is set to `na.exclude`.

This version should be compatiable with the **RExcel** package, which can use the R Commander menus.

Author(s)

John Fox

References

Fox, J. (2017) *Using the R Commander: A Point-and-Click Interface for R*. Chapman and Hall/CRC Press.

Fox, J. (2005) The R Commander: A Basic Statistics Graphical User Interface to R. *Journal of Statistical Software*, **14(9)**: 1–42.

Fox, J. (2007) Extending the R Commander by "plug in" packages. *R News*, **7(3)**: 46–52.

See Also

Plugins, Rcmdr.Utilities, `knit`, `knit2pdf`

Examples

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

`Commander-es` *R Commander*

Description

Inicia la GUI (Interfaz Gráfica de Usuario) de R Commander

Usage

```
Commander()
```

Details

Empezando

La interfaz por defecto de R Commander consiste en (de arriba a abajo) una barra de menús, una barra de herramientas, una ventana de instrucciones, una ventana de salida y una ventana de mensajes.

Las instrucciones para leer, escribir, transformar y analizar datos se ejecutan usando la barra de menú de la parte superior de la ventana de *R Commander*. La mayor parte de los items de este menú le guiarán mediante ventanas de diálogo, preguntando más allá de la especificación. Es aconsejable explorar el menú para ver las opciones disponibles.

Bajo la barra de menú se encuentra la barra de herramientas con (de izquierda a derecha) un campo de información que muestra el nombre del conjunto de datos activos, botones para editar y mostrar el conjunto de datos activos y un campo de información mostrando el modelo estadístico activo. Bajo la ventana de instrucciones hay un botón `Ejecutar` para realizar las órdenes indicadas en la ventana de instrucciones. Los campos de información para los datos y el modelo activo son botones que pueden ser usados para seleccionar éstos entre, respectivamente, conjuntos de datos o modelos disponibles en memoria.

La mayor parte de las órdenes requiere un conjunto de datos activos. Cuando se ejecuta R Commander no hay conjunto de datos activos, como está indicado en el campo de información del conjunto

de datos activos. Un conjunto de datos llega a ser un conjunto de datos activos cuando éste es leído en la memoria desde un paquete R o importado desde un archivo de texto, conjunto de datos SPSS, conjunto de datos Minitab, conjunto de datos STATA, Excel, Access o dBase. Además el conjunto de datos activos puede ser seleccionado desde conjuntos de datos R residentes en memoria. Los datos pueden ser elegidos de entre todos los conjuntos para cada sesión.

Por defecto, las órdenes son registradas en la ventana de instrucciones (la ventana de texto vacía inmediatamente después de la barra de herramientas); las órdenes y las salidas aparecen en la ventana de resultados (la ventana de texto vacía después de la ventana de instrucciones) y el conjunto de datos activos es adjuntado a la ruta de búsqueda. Para alterar éstos y otros parámetros por defecto, puede consultar la información pertinente en configuración.

Algunos diálogos de `Rcmdr` (éstos en `Estadisticos -> Ajuste de modelos`) generan el modelo lineal, modelo lineal generalizado y otros modelos. Cuando un modelo es ajustado, se convierte en el modelo activo, indicado en el campo de información de la barra de herramientas de R Commander. Los items del menú `Modelos` se aplican al modelo activo. Inicialmente, no hay modelo activo. Si hay varios modelos en memoria, puede elegir el modelo activo de entre ellos.

Si el registro de instrucciones está activo, las órdenes de R generadas desde los menús y los cuadros de diálogos, son introducidas en la ventana de instrucciones de R Comander. Se pueden editar estas órdenes de manera normal y se pueden escribir otras nuevas en la ventana de instrucciones. Las órdenes individuales pueden ser continuadas en más de una línea, pero cada línea después de la primera debe ser identada con uno o más espacios o tabuladores. El contenido de la ventana de instrucciones puede ser almacenado durante o al final de la sesión y un conjunto de instrucciones guardado puede ser cargado en la ventana de instrucciones. El contenido de la ventana de resultados puede ser editado o guardado en un archivo de texto.

Para volver a ejecutar una orden o un conjunto de ellas, se seleccionan las líneas que se desean ejecutar usando el ratón y se presiona el botón `Ejecutar`, situado a la derecha de la barra de herramientas (o `Control-R`, para ejecutarlos). Si no hay texto seleccionado el botón `Ejecutar` (o `Control-R`) envía el contenido de la línea que contiene el cursor de insercción. Observar que se generará un error si la orden o las órdenes enviadas son incompletas.

Presionando `Control-F` se abre un cuadro de diálogo de búsqueda de texto (también es accesible vía `Editar -> Buscar`) para buscar el texto en la ventana de instrucciones o la ventana de resultados. Las búsquedas son realizadas en la ventana de instrucciones a menos que primero pulse en la ventana de resultados para activarla.

Presionando `Control-S` se guardará el conjunto de instrucciones o la ventana de resultados.

Presionando `Control-A` se selecciona todo el texto del conjunto de instrucciones o de la ventana de resultados.

Pulsando el botón derecho del ratón (el tercer botón en un ratón de tres botones) en el conjunto de instrucciones o en la ventana de resultados se abre el menú contextual con los items del menú `Editar`, más un item `Ejecutar` (en la ventana de instrucciones).

Cuando ejecute órdenes en la ventana de R Commander, debe asegurarse que la sentencia sea lógica. Por ejemplo, no tiene sentido ajustar un modelo estadístico de un conjunto de datos que no ha sido leído en memoria.

Presionando una letra (ej. "a") en un cuadro con una lista se recorrerá ésta hasta la siguiente entrada que comience con esa letra desde el principio del cuadro.

Salir de R Commander se realiza mediante `Fichero -> Salir` o cerrando la ventana de R Commander.

Personalización y configuración

Los archivos de configuración están en el subdirectorio `etc` de cada paquete o en la localización dada por `etc` y en las opciones de `etcMenus` (mirar abajo).

Los menús de `Rcmdr` pueden ser personalizados editando el archivo `Rcmdr-menus.txt`.

Algunas funciones (ej. `histograma`) que normalmente no crean salida visible cuando se ejecutan desde la consola sí lo harán - a menos que se evite - cuando se ejecuten desde la ventana de instrucciones de R Commander. Tal salida puede ser suprimida listando los nombres de estas funciones en el archivo `log-exceptions.txt`.

Puede añadir código R al paquete, ej., para crear diálogos adicionales, colocando archivos con extensión `.R` en el directorio `etc`, además puede editar `Rcmdr-menus.txt` para proporcionar menús adicionales, submenús o items. Una demostración de esto se proporciona mediante el archivo `BoxCox.demo`. Para activar la demo, renombre el archivo a `BoxCox.R` y descomente la correspondiente línea del menú en `Rcmdr-menus.txt`. De forma alternativa, puede editar el código del paquete y recompilarlo.

Algunas funciones son proporcionadas para ayudar a escribir diálogos y la información del estado de `Rcmdr` en un emplazamiento separado. Mirar `help("Rcmdr.Utilities")` y el manual suministrado en el directorio `doc` del paquete de `Rcmdr` para mayor información.

Además, varias características son controladas mediante opciones, en tiempos de ejecución, establecidas por la orden `options("Rcmdr")`. Estas opciones deben ser establecidas antes de cargar el paquete. Si las opciones no están establecidas, que es la situación normal, serán usados los parámetros por defecto. Las opciones se especifican como una lista de pares `name$=$values`. Puede no establecer, establecer una, varias, o todas las opciones. Las opciones disponibles son las dadas a continuación:

`attach.data.set` Si es `TRUE` (por defecto `FALSE`), el conjunto de datos activo es fijado como la ruta de búsqueda.

`check.packages` Si es `TRUE` (por defecto), al arranque, la presencia de todos los paquetes recomendados de Rcmdr serán comprobados y si alguno no está instalado, Rcmdr preguntará si deben instalarse.

`command.text.color` El color de las órdenes en la ventana de resultados es, por defecto, `"red"`.

`console.output` Si es `TRUE` la salida será dirigida a la consola de R y la ventana de salida de R Commander no se mostrará. Por defecto es `FALSE`.

`contrasts` Ofrece la misma función que la opción general `contrasts`; por defecto es `c("contr.Treatment", "contr.poly")`. Cuando se sale de Commander la opción `contrasts` vuelve a su valor preexistente. Observe que `contr.Treatment` es del paquete `car`.

`crisp.dialogs` Si es `TRUE`, los diálogos deben aparecer en la pantalla dibujada completamente, más que acumular dispositivo a dispositivo. Esta opción debería afectar sólo a versiones Windows de R, pero debe en cualquier caso ser inofensivo. Por defecto es `TRUE` bajo versiones Windows de R 2.1.1 y superiores y `FALSE` si no. Si está trabajando en Windows y encuentra que se incrementan los problemas de estabilidad, pruebe establecer esta opción a `FALSE`.

`default.font` La fuente por defecto, como la especificación de la fuente de X11, dada en cadena de caracteres. Si está especificado, este valor toma precedencia sobre el tamaño de la fuente por defecto (abajo). Esta opción es sólo para sistemas no-Windows.

`default.font.size` Tamaño, en puntos, por defecto de la fuente. Por defecto es 10 para sistemas Windows y 12 para otro sistemas, salvo especificación de lo contrario (mirar el item anterior).

La fuente por defecto es "*helvetica-medium-r-normal-*-xx*", donde xx es por defecto el tamaño de la fuente. Esta opción es sólo para sistemas no-Windows.

double.click Establecer a TRUE si quiere que un doble click con el botón izquierdo del ratón sirva para pulsar el botón por defecto en todos los diálogos. Por defecto es FALSE.

error.text.color Color de los mensajes de error; por defecto es "red".

etc Establece la ruta del directorio que contiene los archivos de configuración de Rcmdr; por defecto el subdirectorio etc del paquete Rcmdr instalado.

grab.focus Establecer a TRUE para "mantener" el enfoque en la ventana actual de Tk, esto es, para prevenir que el enfoque sea cambiado a otra ventana Tk. En algunos sistemas, mantener el enfoque de esta forma, puede causar problemas. Por defecto es TRUE. Si experimenta problemas de enfoque, intente establecer esta opción a FALSE.

load.at.startup Vector de caracteres de nombres de los paquetes que deben ser cargados cuando el paquete Rcmdr es cargado; por defecto se carga sólo el paquete car. Otros paquetes requeridos serán cargados cuando se necesiten. Si esto está disponible, el paquete car será cargado cuando Commander se inicie en cualquier caso.

log.commands Si es TRUE (por defecto), los comandos son repetidos en la ventana de instrucciones; si es FALSE, la ventana de instrucciones no se muestra.

log.font.size Tamaño de la fuente, en puntos, que es usado en la ventana de instrucciones, en la ventana de resultados, en diálogos recodificados y en expresiones de cáculo, esto es, donde es usada una fuente monoespacio. Por defecto es 10 para sistemas Windows y 12 para otros sistemas.

log.height La altura de la ventana de instrucciones, en líneas. Por defecto es 10. Estableciendo log.height a 0 tiene el mismo efecto que establer log.commands a FALSE.

log.text.color Color del texto de la ventana de instrucciones; por defecto es "black".

log.width La anchura de la ventana de instrucciones y la de salida, en caracteres. Por defecto es 80.

multiple.select.mode Afecta a la forma en la que múltiples variables son seleccionadas en una caja de listas de variables. Si se establece a "extended" (por defecto), el botón izquierdo en una variable selecciona ésta y deselecciona cualquier otra variable que estuviera seleccionada; Control+botón izquierdo acciona la selección (y puede ser usado para seleccionar variables adicionales); Mayúsculas+botón izquierdo extiende la selección. éste es el convenio estándar de Windows. Si lo establece a "multiple", el botón izquierdo acciona la selección de una variable y puede ser usado para seleccionar más de una variable. éste es el comportamiento de Rcmdr antes de la versión 1.9-10.

output.height Altura de la ventana de resultados, en líneas. Por defecto es dos veces la altura de la ventana de instrucciones o 20 si la ventana de instrucciones es suprimida. Establecer output.height a 0 tiene el mismo efecto que console.output a TRUE.

output.text.color Color de la salida en la ventana de resultados, por defecto es "blue".

placement Emplazamiento de la ventana de R Commander, en píxeles; por defecto es "$-40+20$", lo que pone la ventana cerca de la esquina superior derecha de la pantalla.

plugins Vector de caracteres con los nombres de paquetes de plugins de Rcmdr a cargar cuando Commander arranque. Los paquetes plugins también pueden ser cargados desde el menú Herramientas -> Cargar paquete(s).

`suppress.menus` Si es `TRUE`, la barra de menús y de herramientas de R Commander son suprimidas, permitiendo que otro programa (como Excel) asuma esas funciones. Por defecto (por supuesto) es `FALSE`.

`suppress.X11.warnings` En (algunos) sistemas Linux X11 se generan múltiples advertencias por las órdenes de Rcmdr, después de abrir la ventana del dispositivo gráfico. Establecer esta opción a `TRUE` (por defecto cuando arranca interactivamente bajo X11 antes de la versión de R 2.4.0) suprime la aparición de estas advertencias. Un efecto secundario indeseable es que entonces todas las advertencias y mensajes de error son interceptados por Rcmdr, incluso para las instrucciones introducidas en los avisos de R. Los mensajes producidos por tales órdenes serán impresos en la ventana de mensajes de R Commander después de la siguiente orden generada en Rcmdr. Algunas advertencias de X11 puede ser impresas al salir de R Commander. Este problema sólo se aplica a versiones de R anteriores a 2.4.0 y el valor por defecto de la opción es establecido por consiguiente.

`retain.messages` Si es `TRUE` (por defecto `FALSE`), el contenido de la ventana de mensajes no es borrado entre mensajes. En cualquier caso, un mensaje "NOTE" no borrará un anterior "WARNING" o "ERROR".

`RExcelSupport` Establecido como `TRUE` (por defecto es `FALSE`), los menús y salidas son dirigidas a Excel.

`scale.factor` Factor de escala aplicado a todos los elementos Tk, como las fuentes. Esto funciona bien sólo en Windows. Por defecto es `NULL`.

`showData.threshold` Si el número de variables en el conjunto de datos activos excede este valor (por defecto, 100), entonces `edit()`, más que `showData()`, es utilizado para exhibir el conjunto de datos. Un inconveniente es que el control no se devuelve a Commander hasta que la ventana de edición sea cerrada. La razón de esta opción es que `showData()` es muy lento cuando el número de variables es grande; fijando el umbral a 0 suprime el uso en conjunto de showData.

`show.edit.button` Fijar a `TRUE` (por defecto) si quiere un botón `Editar` en la ventana de Commander, que permita editar el conjunto activo de datos. Los usuarios de Windows pueden desear establecer esta opción a `FALSE` para suprimir el botón `Editar` porque cambiando los nombres de las variables en el editor de datos se puede causar que R falle (aunque este problema se cree solucionado).

`sort.names` Fijar a `TRUE` (por defecto) si se quiere ordenar alfabéticamente el nombre de las variables en una lista de variables.

`tkwait` Esta opción trata un problema que, en mi conocimiento, es raro y puede ocurrir en algunos sistemas no Windows. Si R Commander causa que se cuelgue R, entonces establezca la opción `tkwait` a `TRUE`; o conserve la opción en `FALSE` e ignórelo. Un indeseable efecto secundario de establecer la opción `tkwait` a `TRUE` es que el aviso de órdenes de la sesión de R es suprimido hasta salir de R Commander. Uno sin embargo todavía puede introducir órdenes por la ventana de instrucciones. En particular, no hay razón para usar esta opción bajo Windows y no se debería usar con la GUI de R en Windows con salida protegida cuando la salida esté dirigida a la consola de R.

`use.rgl` Si es `TRUE` (por defecto), el paquete `rgl` será cargado si está presente en una librería accesible, si es `FALSE`, el paquete `rgl` será ignorado aunque esté disponible. El paquete rgl puede a veces causar problemas cuando se arranca R bajo X11.

`warning.text.color` Color de los mensajes de advertencia; por defecto es `"darkgreen"`.

Muchas opciones pueden también ser establecidas mediante el menú *Archivo -> Opciones*, que reiniciará R Commander después de que las opciones sean establecidas.

Si quiere lanzar R Commander cuando inicie R, puede incluir la siguiente instrucción en uno de los ficheros de inicio de R (por ejemplo, en el fichero `Rprofile.site` de la carpeta `etc` de R):

```
local({
old <- getOption("defaultPackages")
options(defaultPackages = c(old, "Rcmdr"))
})
```

Las opciones de R Commander puede ser establecidas de forma permanente de la misma forma. Para más información sobre el inicio de R, véase `?Startup`.

Avisos

La ventana de instrucciones de R Commander no proporciona una verdadera consola a R y tiene ciertas limitaciones. No se recomienda usar R Commander para la programación sería o el análisis de datos que confíe primordialmente en instrucciones - usar un editor de programación en su lugar. Por ejemplo, para declaraciones de composiciones de R incluidas entre llaves `"\{ \}"`, incluyendo definición de funciones, no serían analizadas ni ejecutadas correctamente, aunque si las líneas después de las primeras que estén identandas. Puede ejecutar declaraciones de composiciones desde la ventana de instrucciones separando los comandos dentro de las llaves por puntos y comas.

Problemas Conocidos

Ocasionalmente, bajo Windows, después de teclear algún texto en un cuadro de diálogo (ej. subconjunto de expresiones en el diálogo de subconjunto de conjunto de datos), algunos botones en el diálogo (ej. el botón Aceptar) pueden no tener efecto cuando sean presionados. Pulsando en cualquier sitio, dentro o fuera del cuadro de diálogo, debería restaurarse las funciones de los botones. Por lo que se ha podido comprobar, éste es un problema con Tcl/Tk de Windows.

Note

Bajo Windows, el paquete `Rcmdr` puede también funcionar bajo de Rgui en modo SDI (interfaz de único documento) o bajo `rterm.exe`; puede ser que experimente problemas ejecutando `Rcmdr` bajo `ESS` con NTEmacs o XEmacs.

Author(s)

John Fox (de la versión inglesa)

Manuel Muñoz Márquez (traductor–mantenedor) `<manuel.munoz@uca.es>`

Véase https://knuth.uca.es/R/doku.php?id=equipotraduccion

See Also

Plugins

Examples

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

Compute *Rcmdr Compute Dialog*

Description

Rcmdr Compute Dialog

Usage

```
Compute()
```

Details

The compute dialog is used to compute new variables.

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

See Also

Arithmetic

editDataset *R Commander Dataset Editor*

Description

R Commander Dataset Editor

Usage

```
editDataset(data, dsname, ...)

## S3 method for class 'character'
editDataset(data, dsname, ...)

## S3 method for class '`NULL`'
editDataset(data, dsname, ...)

## S3 method for class 'data.frame'
editDataset(data, dsname, ...)
```

Arguments

data	an R data frame to edit; this argument is optional, and if absent an empty data frame is created, into which the user can enter data.
dsname	the quoted name of the data set, into which the edited data frame will be placed in the global environment. If absent and an *existing* data frame is edited, the modified version will replace the original version; if absent and a *new* data set is created, it will be given the name "Dataset".
...	not used by the data.frame method.

Details

Allows the user to enter a new dataset, modify data values in an existing dataset, add rows or columns to the dataset, or delete rows or columns.

editDataset is a straightforward spreadsheet-like data editor, suitable for editing data frames that are not too large (say smaller than about 10,000 values). It is defined as a generic function with a data.frame method to allow for objects with unique properties that inherit from the data.frame class. The character and NULL methods permit editing an initially empty data set.

- Use the mouse and the arrow keys to navigate the cells of the data table, including the row and column names.
- Columns consisting only of numbers will produce numeric variables in the data frame constructed by editDataset; columns with any non-numeric values will produce factors or (if they contain only the values TRUE and FALSE) logical variables.
- When entering values with embedded blanks, it is permissible but not necessary to enclose the values in quotes (e.g, "some PS" or 'less than HS').
- Clicking in a cell and typing a new value replaces the previous value.
- Row and column names can be modified in the same manner.
- Double-clicking in a cell deletes the previous value and replaces it with NA.
- Enlarge the data set by pressing the *Add row* or *Add column* button at the top of the data editor; the new row or column will initially be filled with NAs and will have an auto-generated row or column name. Pressing the *Enter* or *Return* key will also add a row; pressing the *Tab* key will also add a column.
- Right-clicking (or *Control*-clicking, or under Mac OS X *Command*-clicking) brings up a context menu, permitting several operations on cells, rows, and columns, including deleting the current row or column.
- Similarly, several actions are available via the *Edit* menu.
- The key-combinations *Control-x*, and *Control-v*, may also be used respectively to cut, copy, and paste cell values. (Under Mac OS X, *Command-x*, *Command-c*, and *Command-v* also work.)
- Pressing the *OK* button or selecting *Exit and save* from the *File* menu exits the data edtior and saves the edited data set to the global environment. Pressing the *Cancel* button or selecting *Cancel* from the *File* menu exits the editor discarding the edited data set.

Value

This function does not return a useful value, but has the side effect of modifying or creating a data set in the global environment.

Note

`editDataset` is limited to editing data frames that are composed only of numeric, factor, and logical columns.

Author(s)

John Fox

See Also

`edit.data.frame`, for the standard R data editor.

Examples

```
if (interactive()) editDataset()
```

`generalizedLinearModel`

Rcmdr Generalized Linear Model Dialog

Description

Rcmdr Generalized Linear Model Dialog

Usage

```
generalizedLinearModel()
```

Details

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

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 selected) 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. If you select several variables in the variable-list box, clicking on the `+`, `*`, or `:` button will enter them into the model 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.

The weights box allows you to select a variable specifying prior weights from the drop-down list. Weights giving numbers of trials may be used, for example, to fit a binomial GLM; in this case, the response variable should give the proportion of "successes" for each binomial observation. Click in the weights combo box to see a list of numeric variables in the current data set; type a letter in the box to move the selection cursor to the next variable beginning with that letter.

There is an optional case-deletion box, whose presence is controlled by the `model.case.deletion` R Commander option (see `Commander`). Typing the row numbers (e,g., `6 16`) or row names (e.g., `minister conductor`) of cases to be deleted removes these cases from the fitted linear model. Row names with embedded blanks must be quoted (e.g., `"railroad engineer"`), in which case *all* row names specified should be quoted (e.g., `"railroad engineer" "minister"`). You cannot specify *both* a subset expression and case deletion.

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, weights, and subset fields are retained from the active model.

Author(s)

John Fox

See Also

`glm`, `family`, `Comparison`

---

hierarchicalCluster *Rcmdr Hierarchical Clustering Dialog*

Description

Rcmdr Hierarchical Clustering Dialog

Usage

```
hierarchicalCluster()
```

Details

This dialog is used to specify a hierarchical cluster analysis solution using `hclust`, with the distance matrix calculated using `dist`.

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 gener 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`

`linearModel` *Rcmdr Linear Model Dialog*

Description

Rcmdr Linear Model Dialog

Usage

```
linearModel()
```

Details

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

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 selected) or to the right-hand side. You can also enter operators and parentheses using the buttons above the formula. If you select several variables in the variable-list box, clicking on the `+`, `*`, or `:` button will enter them into the model 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.

The weights box allows you to perform weight-least-squares (WLS) regression; select a weight variable from the drop-down list. Click in the weights combo box to see a list of numeric variables in the current data set; type a letter in the box to move the selection cursor to the next variable beginning with that letter.

There is an optional case-deletion box, whose presence is controlled by the model.case.deletion R Commander option (see Commander). Typing the row numbers (e,g., 6 16) or row names (e.g., minister conductor) of cases to be deleted removes these cases from the fitted linear model. Row names with embedded blanks must be quoted (e.g., "railroad engineer"), in which case *all* row names specified should be quoted (e.g., "railroad engineer" "minister"). You cannot specify *both* a subset expression and case deletion.

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, weights, and subset fields are retained from the previous model.

Author(s)

John Fox

See Also

lm, Comparison

Plugins *R Commander Plug-in Packages*

Description

R Commander Plug-in Packages

Usage

```
listPlugins(loaded = FALSE)
```

Arguments

loaded	if TRUE, plug-in packages that are loaded are included in the vector of names returned.

Details

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

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 an RcmdrModels: field in the package's DESCRIPTION

file, augments 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. It is also possible to remove menus and menu items from the standard Commander menu file or from the files of plug-ins installed before the current one.

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.

A variety of utility functions is available to support R Commander plug-in packages; see `Rcmdr.Utilities`.

For more details, see Fox, *Writing R Commander Plug-in Packages* at `https://www.john-fox.ca/RCommander/plug-ins.html`.

See Also

`Commander`, `Rcmdr.Utilities`.

`Rcmdr.Utilities` *Rcmdr Utility Functions*

Description

Funtions to support additions to the Rcmdr.

These functions support writing additions to the Rcmdr package, preferably by writing an Rcmdr plug-in package. Although it is not recommended, additional R code can also be placed in files with file type `.R` in the `etc` subdirectory of the **Rcmdr** package. In this case, you can add menus, submenus, and menu items by editing the file `Rcmdr-menus.txt` in the same directory.

`listVariables` retrieves the variable names from the specified `dataSet` or from the active data set.

If called without arguments, the function returns the names of the variables in the active data set.

If the `dataSet` argument is provided, the function returns the variable names associated with that specific data set.

`listPositiveVariables` is a specialized version of `listVariables` that works only with variables containing only positive values.

`radioButtons` creates a group of radio buttons within a Tk dialog, allowing the user to select a single option from a predefined list.

This function generates a set of related radio buttons. It supports "dialog memory," meaning it can retain and restore the user's previous selections across different invocations of the same dialog.

`Variables` retrieves the variable names from the active data set or stores a new set of names.

If called without arguments, the function returns the names of the variables in the active data set.

If the `names` argument is provided, the function stores these as the current variable names for the active session.

NumericPositive retrieves or stores the names of variables containing only positive values in the active data set.

This function is a specialized version of Variables designed specifically to work with positive-valued variables.

numericPositiveP is a specialized version of numericP that checks if a variable is both numeric and contains strictly positive values.

varPosn returns the indices of specific variables within the active data set, filtered by type and sorted alphabetically.

varPosn supports "dialog-box memory" (i.e., retaining selections across successive invocations of a dialog).

Usage

```
activateMenus()
activeDataSet(dsname, flushModel=TRUE, flushDialogMemory=TRUE)
ActiveDataSet(name)
activeDataSetP()
activeModel(model)
ActiveModel(name)
activeModelP()
anovaP()
beginRmdBlock()
beginRnwBlock()
Character(names)
characterP(n=1)
checkActiveDataSet()
checkActiveModel()
checkBoxes(window=top, frame=stop("frame not supplied"),
    boxes=stop("boxes not supplied"),
    initialValues=NULL, labels=stop("labels not supplied"),
    title=NULL, ttk=FALSE, columns=1)  # 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
Coef(object, ...)
Predictors(type=c("all", "numeric", "factor"))
PredictorsP(n=1, type=c("all", "numeric", "factor"))
CommanderWindow()
dataSetsP(n=1)
defmacro(..., expr)
```

```
dialogSuffix(window=top, onOK=onOK, onCancel=onCancel, rows, columns,
focus=top, bindReturn=TRUE,
    preventGrabFocus=FALSE, preventDoubleClick=FALSE, preventCrisp,
    use.tabs=FALSE, notebook=notebook, tabs=c("dataTab", "optionsTab"),
    tab.names=c("Data", "Options"), grid.buttons=FALSE, resizable=FALSE,
    force.wait=FALSE)  # macro
DiscreteNumeric(names)
discreteNumericP(n=1)
doItAndPrint(command, log=TRUE, rmd=log)
EffectP()
endRmdBlock()
endRnwBlock()
enterMarkdown(command)
enterKnitr(command)
errorCondition(window=top, recall=NULL, message, model=FALSE)  # macro
exists.method(generic, object, default=TRUE, strict=FALSE)
Factors(names)
factorsP(n=1)
formulaFields(model, hasLhs=TRUE, glm=FALSE)
flushDialogMemory(what)
gassign(x, value)
getCases(cases, remove=TRUE)
getDialog(dialog, defaults=NULL)
## S3 method for class 'combobox'
getFrame(object)
## S3 method for class 'listbox'
getFrame(object)
## S3 method for class 'combobox'
getSelection(object)
## S3 method for class 'listbox'
getSelection(object)
getRcmdr(x, mode="any", fail=TRUE)
gettextRcmdr(...)
glmP()
GrabFocus(value)
groupsBox(recall=NULL, label=gettextRcmdr("Plot by:"),
    initialLabel=gettextRcmdr("Plot by groups"),
    errorText=gettextRcmdr("There are no factors in the active data set."),
    variables=Factors(),
    plotLinesByGroup=FALSE, positionLegend=FALSE,
    plotLinesByGroupsText=gettextRcmdr("Plot lines by group"),
    initialGroup=NULL, initialLinesByGroup=1, window=top)  # macro
groupsLabel(frame=top, groupsBox=groupsBox, columnspan=1,
    initialText=NULL, ratio=FALSE)  # macro
hclustSolutionsP()
initializeDialog(window=top, title="", offset=10, preventCrisp,
    use.tabs=FALSE, notebook=notebook,
    tabs=c("dataTab", "optionsTab"),
```

```
    suppress.window.resize.buttons=TRUE)  # macro
insertRmdSection(text)
is.valid.name(x)
is.valid.number(string)
is.SciViews()
justDoIt(command)
knitrP()
Library(package, pos=length(search()), rmd=TRUE)
listAllModels(envir=.GlobalEnv, ...)
listAOVModels(envir=.GlobalEnv, ...)
listCharacter(dataSet=ActiveDataSet())
listDataSets(envir=.GlobalEnv, ...)
listDiscreteNumeric(dataSet=ActiveDataSet())
listFactors(dataSet=ActiveDataSet())
listGeneralizedLinearModels(envir=.GlobalEnv, ...)
listLinearModels(envir=.GlobalEnv, ...)
listMultinomialLogitModels(envir=.GlobalEnv, ...)
listNumeric(dataSet=ActiveDataSet())
listProportionalOddsModels(envir=.GlobalEnv, ...)
listTwoLevelFactors(dataSet=ActiveDataSet())
listVariables(dataSet=ActiveDataSet())
lmP()
logger(command, rmd=TRUE)
logLikP()
LogWindow()
MacOSXP(release)
manualTranslationP()
MarkdownP()
mavericksP()
Message(message, type=c("note", "error", "warning"))
MessagesWindow()
modelCapability(capability)
modelFormula(frame=top, hasLhs=TRUE, rhsExtras=NULL,
    formulaLabel=gettextRcmdr("Model Formula"), showBar=FALSE)  # macro
modelsP(n=1)
multinomP()
Numeric(names)
numericP(n=1)
OKCancelHelp(window=top, helpSubject=NULL, model=FALSE,
    reset=NULL, apply=NULL, helpPackage=NULL)  # macro
OutputWindow()
packageAvailable(name)
polrP()
popCommand(keep=FALSE)
popOutput(keep=FALSE)
putDialog(dialog, values=NULL, resettable=TRUE)
putRcmdr(x, value)
RappP()
```

```
RcmdrEditor(buffer, title="R Commander Editor", ok,
    help=NULL, file.menu=NULL, edit.menu=NULL, context.menu=NULL,
    toolbar.buttons=NULL)
RcmdrTclSet(name, value)
RcmdrTkmessageBox(message, icon=c("info", "question", "warning",
    "error"), type=c("okcancel", "yesno", "ok"), default, title="")
removeLastRmdBlock()
removeLastRnwBlock()
removeNullRmdBlocks()
removeNullRnwBlocks()
removeStrayRmdBlocks()
removeStrayRnwBlocks()
RExcelSupported()
rglLoaded()
RmdWindow()
RnwWindow()
setBusyCursor()
setIdleCursor()
sortVarNames(x)
subOKCancelHelp(window=subdialog, helpSubject=NULL)  # macro
subsetBox(window = top, subset.expression = NULL, model = FALSE)  # macro
suppressMarkdown(command)
tclvalue(x)
titleLabel(...)
tkspinbox(parent, ...)
trim.blanks(text)
TwoLevelFactors(names)
twoLevelFactorsP(n=1)
UpdateModelNumber(increment=1)
variableComboBox(parentWindow, variableList=Variables(),
    export="FALSE", state="readonly",
    initialSelection=gettextRcmdr(nullSelection),
    title="", nullSelection="<no variable selected>",
    adjustWidth = FALSE)
variableListBox(parentWindow, variableList=Variables(), bg="white",
    selectmode="single", export="FALSE", initialSelection=NULL,
    listHeight=getRcmdr("variable.list.height"), title)
Variables(names)
WindowsP()
X11P()
commanderPosition()

listVariables(dataSet = ActiveDataSet())

listNumericPositive(dataSet = ActiveDataSet())

radioButtons(
  window = top,
```

```
  name = stop("name not supplied"),
  buttons = stop("buttons not supplied"),
  values = NULL,
  initialValue = ..values[1],
  labels = stop("labels not supplied"),
  title = "",
  title.color = getRcmdr("title.color"),
  right.buttons = FALSE,
  command = function() {
 },
  columns = 1
)

Variables(names)

NumericPositive(names)

numericPositiveP(n = 1)

varPosn(
  variables,
 type = c("all", "factor", "numeric", "numericpositive", "nonfactor", "twoLevelFactor"),
  vars = NULL
)
```

Arguments

dataSet	An optional character string specifying the name of a data set. If omitted, the function defaults to the active data set.
window	A Tk window object serving as the parent container.
name	A character string providing the base name for the widget.
buttons	A character vector of names for the individual radio buttons in the set.
values	A character vector of values associated with each radio button. For putDialog, this should be a list of current selections to be stored in the dialog's memory.
initialValue	The value of the radio button that should be selected by default.
labels	A character vector of strings to be used as labels for the radio buttons.
title	A character string for the title of the radio button group.
title.color	The color of the title text; defaults to "blue".
right.buttons	Logical; if TRUE, the buttons are placed to the right of their labels. Defaults to FALSE.
command	A character string that evaluates to an R command, or a function to be executed when a button is selected.
columns	An integer (1, 2, 3, or 4) specifying the number of columns used to arrange the buttons. Buttons are filled by rows; defaults to 1.
names	An optional character vector of variable names to be stored. If omitted, the function defaults to retrieval mode.

n	number of items to check for.
variables	A character vector containing one or more variable names.
type	A character string specifying the type of object to check. Used to filter the variable list for varPosn.
vars	A character vector of variable names. If provided, the type argument is ignored.
adjustWidth	adjust width of combo box to accommodate widest entry (default FALSE).
ask	ask for confirmation.
ask.save	ask whether to save contents of script and output windows.
apply	if non-null (the default is NULL), an Apply button is included in the dialog's button bar. This argument should be set to the quoted name of the function that initiates the dialog; when the button is pressed, the onOK function for the dialog is executed, and then the function named in apply is (re)called.
bg	background color.
bindReturn	if TRUE, the *Return* key is bound to the onOK function in the dialog.
boxes	vector of quoted names for check boxes, used to generate each box and its associated variable.
buffer	a text string, typically representing the contents of a text widget, such as an R Markdown or knitr document.
capability	character string giving the name of a column in the R Commander model-capabilities table, including the name of a column added by a plug-in; see model-capabilities.txt in the **Rcmdr** sources for the standard table; e.g., "sum" indicates the availability of an applicable summary() method for the current model.
cases	a character string of case number or names to be removed or retained, separated by blanks.
class	quoted name of class.
columnspan	number of dialog-box columns to be spanned by frame.
context.menu	NULL or a list containing one or more two-element lists: the first element, label, supplies the text label for a menu item in the RcmdrEditor right-click context menu; the second element, command, is a call-back function to be evaluated when the menu item is selected. If NULL (the default), no item will be added to the file menu.
default	default button: if not specified, "ok" for "okcancel", "yes" for "yesno", and "ok" for "ok"; or look for a default method; for putDialog, a list of defaults for the dialog box if there are no stored previous values.
defaults	a list of named default values for options in a dialog if no previous selections are stored.
dialog	the quoted name of a dialog box under which previous selections are stored.
dsname	name of the data set to activate.
edit.menu	NULL or a list containing one or more two-element lists: the first element, label, supplies the text label for a menu item in the RcmdrEditor Edit menu; the second element, command, is a call-back function to be evaluated when the menu item is selected. If NULL (the default), no item will be added to the file menu.

envir	the environment to be searched; should generally be left at the default.
errorText	error message to print if a suitable factor isn't available.
export	export selection?
expr	An expression constituting the body of the macro; typically a compound expression.
fail	if TRUE, the default, getRcmdr will generate an error if the object sought doesn't exist; if FALSE and the object doesn't exist, NULL is returned.
file.menu	NULL or a list containing one or more two-element lists: the first element, label, supplies the text label for a menu item in the RcmdrEditor File menu; the second element, command, is a call-back function to be evaluated when the menu item is selected. If NULL (the default), no item will be added to the file menu.
flushDialogMemory	remove saved values of dialog options so that getDialog returns NULL for all dialogs.
flushModel	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.
focus	Tk window to get the focus.
force.wait	call tkwait.window so that processing is suspended until the dialog is closed; overrides the **Rcmdr** tkwait.dialog option (see Commander) if the latter is set to FALSE (its default). The force.wait argument should be set to TRUE for subdialogs.
formulaLabel	text label printed above the formula widget.
frame	frame or quoted name for frame depending upon the function.
generic	quoted name of generic function.
glm	TRUE if the model is a glm object, FALSE otherwise.
grid.buttons	insert call to tkgrid for the buttons frame (default FALSE); use TRUE for tabbed dialogs and optionally for other dialogs.
groupsBox	listbox object for selecting groups variable.
initialText	initial text to display in the groups label; if NULL, "<No groups selected>" will be displayed.
hasLhs	does the model formula have a left-hand side?
help	a two element list: the first element, label, supplies the text label for a menu item in the RcmdrEditor Help menu; the second element, command, is a call-back function to be evaluated when the menu item is selected. If NULL (the default), no item will be added to the editor Help menu.
helpSubject	the quoted name of a help subject, to be called as help(helpSubject) when the dialog *Help* button is pressed.
helpPackage	the quoted name of the package in which to look for help; the default, NULL, produces a search in all loaded packages — see help.
icon	Message-box icon.
increment	increment to model number; -1 to set back after error.

initialGroup	quoted name of variable to define groups, set as initial selection in Groups variable list; NULL (the default) for no initial selection.
initialLinesByGroup	if 1, the lines-by-groups check box is initially checked; 0 to uncheck.
initialLabel	label for groups button before a selection is made.
initialSelection	index of item initially selected, 0-base indexing.
initialValues	for a set of related check boxes.
keep	if TRUE, keep (rather than pop) last output or command in the stack; the default is FALSE.
label	label prefix for groups button after a selection is made.
listHeight	Maximum number of elements displayed simultaneously in list box.
log	echo command to the script window, as well as executing it and printing its output.
message	error (or other) message.
mode	mode of object to retrieve.
model	the name of a model, as a character string, or a model object, or TRUE or FALSE, depending upon the function.
notebook	notebook windows for a tabbed dialog (default notebook).
nullSelection	what user selects in combo box to indicate nothing selected (default "<no variable selected>").
object	an object (depends on context).
offset	in pixels, from top-left of Commander window.
ok	a function called when the *OK* button is pressed in a RcmdrEditor window that saves the editor buffer to the appropriate *Tcl* variable.
onOK	function to execute when the *OK* button is pressed.
onCancel	function to execute when the *Cancel* button or *Esc* key is pressed.
package	quoted name of package to load.
parent	A Tk window object serving as the parent container.
parentWindow	A Tk window object serving as the parent container.
plotLinesByGroup	include a check box for plotting lines by group?
plotLinesByGroupsText	the label for the plot-lines-by-group check box.
pos	position on search path at which to load package; default is just before the end of the path.
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`
: this argument is ignored, and is present only for backwards compatibility.

`ratio`
: If `FALSE`, the default, the test will be expressed as a test for a difference (e.g., a difference in means); if `TRUE`, as a test for a ratio (e.g., a ratio of variances).

`recall`
: function to call after error — usually the function that initiates the dialog.

`release`
: release the focus if the grab.focus option has been set. In `MacOSXP`, the minimum release (version) required.

`remove`
: If `TRUE`, the default, return a character string that evaluates to an expression of case numbers or names to remove from the data set; if `FALSE`, return a character string that evaluates for an expression of case numbers or names to retain.

`reportError`
: if `TRUE`, report an error message.

`reset`
: quoted name of dialog function, to be invoked with all defaults by Reset button.

`resettable`
: should dialog state be reset when the data set changes? The default is `TRUE`.

`resizable`
: should the dialog be resizable by the user? The default is `FALSE`.

`rhsExtras`
: show controls for splines and polynomials for a model formula; for backwards compatibility, defaults to `TRUE` for a two-sided formula and `FALSE` for a one-sided formula.

`rmd`
: enter the command in the R Markdown tab.

`rows`
: numbers of rows of widgets in the dialog box; this is actually no longer used, but is still present for backwards compatibility. The `columns` argument is similarly ignored, except for radio buttons and check boxes.

`selectmode`
: `"single"` or `"multiple"`.

`showBar`
: include a *bar* (|) button in the model-formula widget (default is `FALSE`).

`state`
: state of the combobox widget; default `"readonly"` means user can't type in the box; set to `"normal"` to permit typing.

`strict`
: if `TRUE`, only use first element of class vector.

`string`
: a character string, or vector of strings, to be tested whether it can be coerced to a number or numbers; returns either `TRUE` or `FALSE`.

`subset.expression`
: a quoted expression to subset the data set.

`suppress.window.resize.buttons`
: if `TRUE`, the default, the window maximize/minimize buttons will not be displayed.

`tab.names`
: text to print as tab labels (default `c("Data", "Options")`).

`tabs`
: quoted names of tabs for a tabbed dialog (default `c("dataTab", "optionsTab")`).

`text`
: a text string.

toolbar.buttons	NULL or a list containing one or more three-element lists: the first element, label, supplies the text label for a button in the RcmdrEditor toolbar; the second element, command, is a call-back function to be evaluated when the button is pressed; the third element is the name of a tk image to display as an icon in the button. If NULL (the default), no buttons will be added to the toolbar.
ttk	use ttk themed widget for check boxes.
use.tabs	(default FALSE) construct a tabbed dialog.
value	an object to be stored or assigned.
variableList	a vector of variable names.
what	optional character vector of one or more dialog names for which the memory is to be flushed; if not specified, all dialog memory will be flushed.
x	an R object name, as a character string, or a tcl variable or object, or a vector of variable names to be sorted.
...	For gettextRcmdr, text string or vector of text strings to translate; for titleLabel, arguments to be passed to labelRcmdr and from there to ttklabel; for defmacro, arguments for the macro; otherwise you should disregard this argument.

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. Also see the **RcmdrPlugin.survival** package for examples.

This function filters the variables in the specified or active data set, returning only those that meet the positivity criterion.

While numericP only verifies the data type, numericPositiveP validates the actual content of the variable. It returns TRUE only if the variable meets both criteria: being numeric and having values strictly greater than zero.

Value

A character vector containing the names of the variables.

This function does not return a value. Instead, it creates a Tcl variable (named as nameVariable) that stores the user's selection.

If names is missing, returns a character vector of variable names. If names is provided, the function is used for its side effect of storing the names (typically returns NULL invisibly).

Returns TRUE if the condition is met, and FALSE otherwise.

A numeric vector representing the 0-indexed positions of the requested variables after filtering the data set by type and sorting alphabetically.

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. The R Commander maintains a list of output objects, by default including the last 10 outputs. `popOutput()` "pops" (i.e., returns and removes) the first entry of the output stack. Note that, as a stack, the queue is LIFO ("last in, first out"). Use `popOutput(keep=TRUE)` to access the last output but keep it in the stack. There is also a stack of commands, which is accessed similarly by `popCommand()`. Ocassionally, it's necessary to assign an object directly in the global environment, and this can be done with the `gassign` function.

Normally commands also generate an R Markdown block. `suppressMarkdown` takes a command in character-string form and adds an attribute to it that will cause the command *not* to be entered in the R Markdown tab. This is useful when a command, such as `identify`, requires direct user interaction and won't generate useful Markdown. `enterMarkdown` can be used to enter command blocks directly in the R Markdown tab; this should rarely be required. The functions `beginRmdBlock`, `endRmdBlock`, `removeNullRmdBlocks`, `removeLastRmdBlock`, and `removeStrayRmdBlocks` should normally not be called directly. The functions `enterKnitr`, `beginRnwBlock`, `endRnwBlock`, `removeNullRnwBlocks`, `removeLastRnwBlock`, and `removeStrayRnwBlocks` perform similar functions for Knitr documents. `insertRmdSection` Inserts a Markdown section title immediately above the last R command block, with the specified text as the title. In most instances it's unnecessary to do this directly because most commands automatically generate a Markdown section title.

Checking for errors

The function `is.valid.name` checks whether a character string specifies a valid name for an R object. The function `is.valid.number` checks whether a character string (or vector) can be coerced to a number (or numbers). 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`, `listAOVModels`, `listCharacter`, `listDataSets`, `listDiscreteNumeric`, `listGeneralizedLinearModels`, `listFactors`, `listLinearModels`, `listMultinomialLogitModels`, `listNumeric`, `listProportionalOddsModels`, `listTwoLevelFactors`, `listVariables`. The functions `Predictors` and `Coefficients` return information about the active model, or `NULL` if there is no active model. 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`. The function `is.SciViews` always returns `FALSE` since the SciViews GUI is no longer supported.

The function `modelCapability()` returns `TRUE` if there is an active statistical model and if it has the specified capability. For example, `modelCapability("sum")` returns `TRUE` if the model-capabilities

table indicates that there's an applicable `summary()` method for the active model. Otherwise, `FALSE` is returned. If the specified capability doesn't exist, a warning is printed.

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 associated methods for the generic functions `getFrame` and `getSelection`); a drop-down "combo" box, `variablecomboBox` (with `getFrame` and `getSelection` methods); 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`. The functions `putDialog`, `getDialog`, and `varPosn` support dialog-box memory—i.e., retaining selections across invocations of a dialog. The `tkspinbox` function is omitted from the tcltk package and may be used to create a spinbox widget. The `titleLabel` function may be used to format a title label to use the standard title label font and color.

"Themed" Tk widgets

Tk 8.5 introduced so-called "themed" widgets, which look better than the traditional Tk widgets. Several functions, contributed by Brian Ripley, are written to access the new widgets by switching automatically between the new and old widget sets depending upon the availability of the former: `buttonRcmdr`, to access either `ttkbutton` or `tkbutton`; `labelRcmdr`, to access either `ttklabel` or `tklabel`; `ttkentry`, to access either `ttkentry` or `tkentry`; `ttkframe`, to access either `ttkframe` or `tkframe`; `ttkradiobutton`, to access either `ttkradiobutton` or `tkradiobutton`; and `ttkscrollbar`, to access either `ttkscrollbar` or `tkscrollbar`. Note that the last four functions mask functions of the same names in the **tcltk** package.

"Predicate" functions

A number of functions of the form *name*`P` are 'predicate' functions, which return `TRUE` or `FALSE` depending upon whether some condition obtains. For example, `lmP()` returns `TRUE` if there is an active model that is a linear model; and `factorsP(2)` returns `TRUE` if there are at least two factors in the active data set. `WindowsP()`, `MacOSXP()`, and `X11P()` return `TRUE` if the R Commander is running under Windows, Mac OS X, or X-Windows, consecutively.

Translating text

The `gettextRcmdr` function simply passes its argument(s) to `gettext`, adding the argument `domain="R-Rcmdr"`. It is not meant to be used in plug-in packages.

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.

The function `nobs` returns the number of observations on which a statistical model is based.

The function `formulaFields` returns information about the left-hand side, right-hand side, data, subset, and (for GLMs) family and link, of a model object.

The function `sortVarNames` sorts variable names, including those containing numerals, into a more "natural" order than does the standard `sort` function.

The function `Library` may be used to load packages; it checks whether a package is already loaded, and if not by default puts it in position 4 on the search path.

The function `Coef`, with several methods, returns the coefficients of a model as a vector; the default method just calls `coef`.

The function `RExcelSupported` is used for the RExcel interface.

The function `getCases` takes a character string of case names or numbers as an argument and returns a character string that evaluates to an expression either to delete or retain these cases.

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), and are used in the R Commander to deal with scoping issues related to Tcl/Tk.

The `tkfocus` function is exported for historical reasons.

Author(s)

John Fox

References

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

`RcmdrPager` *Pager for Text Files*

Description

Pager for Text Files

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.

Details

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

See Also

tkpager.

RecodeDialog *Rcmdr Recode Dialog*

Description

Rcmdr Recode Dialog

Usage

```
RecodeDialog()
```

Details

Recode numeric variables and factors into factors.

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.

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 `lo` and `hi` may appear in a range. For example, `lo: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.

As explained, `=` is used to separate old from new values, and `:` is used to specify an interval (or range) of numeric values. It is possible to change these operators to other character strings, such as `->` and `~` (tilde). This may be necessary, for example, if a factor to be recoded has `=`s or `:`s in its level (category) names.

Similarly, the dialog generates a call to the `Recode` function in the **car** package, which by default uses `;` to separate recode specifications. The recode separator can also be changed, for example, to `/`.

Author(s)

John Fox

See Also

`Recode`

`RepeatedMeasuresDialogs` *R Commander Repeated-Measures ANOVA/ANCOVA Dialogs*

Description

R Commander Repeated-Measures ANOVA/ANCOVA Dialogs

Usage

```
oneWayRepeatedMeasures()
```

Details

One-way and two-way repeated-measures ANOVA/ANCOVA

The one-way and two-way repeated-measures ANOVA/ANCOVA dialogs compute analysis of variance and analysis of covariance tables for one or two repeated-measures factors and a between-subjects linear model that can include both factors and covariates.

The data are assumed to be in "wide" format, with repeated measures corresponding to distinct variables in the active data set. If the data are in "long" format, they can be reshaped to wide format via the *Data* > *Active data set* > *Reshape data set from long to wide format* dialog.

The model is specified in the *Design* tab in two parts:

1. The within-subjects design is defined by using the drop-down lists to select the variables that correspond to the levels of the within-subjects factor (in the case of one repeated-measures factor) or the combinations of levels of the within-subjects factors (in the case of two repeated-measures factors, organized as a two-way table). The user can also name the levels of the within-subjects factor(s) and the factor or factors themselves.
2. The between-subjects model is specified exactly as for a linear model (see `linearModel`). The model is then fit as a multivariate linear model with the repeated measures as response variables.

In the *Options* tab, the user can select either multivariate tests (using one of four test statistics) or univariate tests. The tests are performed by the `Anova` function in the **car** package.

The means of the repeated measures can optionally be plotted as a function of within- and between-subjects factors, and the means and standard deviations can be printed, using the `repeatedMeasuresPlot` function in the **RcmdrMisc** package.

Author(s)

John Fox

References

John Fox, Michael Friendly, and Sanford Weisberg, "Hypothesis Tests for Multivariate Linear Models Using the **car** Package", *The R Journal*, 5 (1) 39–52, 2013.

See Also

`linearModel`, `Anova`, `repeatedMeasuresPlot`.

`ReshapeDatasetDialogs` *R Commander Reshape Data Set Dialogs*

Description

R Commander Reshape Data Set Dialogs

Usage

```
reshapeLong2Wide()
```

Details

Reshape data sets between long and wide formats.

There are two dialogs for "reshaping" the active data set: (1) from "long" to "wide" format (using the `reshapeL2W` in the **RcmdrMisc** package); and (2) from "wide" to "long" format (using the `reshapeW2L` in the **RcmdrMisc** package).

These dialogs are designed for handling regularly structured repeated-measures data, where each subject (independent unit of observation) is measured on several occasions or under several different

conditions. The occasions may have either a one-dimensional structure (corresponding to the levels of one repeated-measures or within-subjects factor) or a two-dimensional structure (corresponding to two crossed repeated-measures factors).

Data in "wide" format have one row for each subject, with the repeated measures in different columns (variables). Data in "long" format have several rows for each subject, with one column (variable) for the response; the levels of the repeated-measures factor (or combinations of levels for two repeated-measures factors) correspond to distinct rows. The within-subjects factor or factors appear as different columns in the long data, as do the between-subjects factors; the former vary within subjects, while the latter are invariant within subjects; and an ID variable identifies the rows of the data set belonging to each subject.

Data in wide format are suitable for analysis by the R Commander repeated-measures ANOVA/ANCOVA dialogs (see `RepeatedMeasuresDialogs`), while data in long format are suitable for analysis by the R Commander mixed-models dialogs.

The *Reshape Data Set from Long to Wide Format* dialog is largely self-explanatory: The user selects the variable that identifies subjects (i.e., the ID variable); one or two within-subjects factors; the variable or variables that vary by occasion (there is typically just one, the response variable); and any variables in the long data set that are to be excluded from the wide data set.

The *Reshape Data Set from Wide to Long Format* dialog is more complex. There are three tabs: A tab to specify one repeated-measures factor; a tab to specify two crossed repeated-measures factors; and an options tab. The user specifies *either* one *or* two repeated-measures factors, not *both*. The within-subjects factor or factors are defined by using drop-down lists to select the variables that correspond to the levels of the within-subjects factor (in the case of one repeated-measures factor) or the combinations of levels of the within-subjects factors (in the case of two repeated-measures factors, organized as a two-way table). The user can also name the levels of the within-subjects factor(s) and the factor or factors themselves.

Author(s)

John Fox

See Also

`reshapeL2W`, `reshapeW2L`, `RepeatedMeasuresDialogs`.

`saveOptions` *Save R Commander Options in an R Profile File*

Description

Save R Commander Options in an R Profile File

Usage

```
saveOptions()
```

Details

This dialog creates a '.Rprofile' file, by default in the current directory, adding to it the current R Commander options, set, e.g., in the Options dialog. If R is subsequently restarted in this directory, and the **Rcmdr** package loaded, then the current R Commander options will be applied. The current directory will typically, though not necessarily, be your home directory — for example, for Windows users, your Documents directory.

If a '.Rprofile' file already exists in the current directory, then the R Commander options are added to it at the end of the file, after removing R Commander options previously generated by an earlier invocation of this dialog.

The contents of the '.Rprofile' file may be edited before you save the file. If you want to start the R Commander automatically when R starts, uncomment (remove the #s from) the four lines

```
# local({
#    old <- getOption('defaultPackages')
#    options(defaultPackages = c(old, 'Rcmdr'))
# })
```

See Startup for a description of the '.Rprofile' file and the R startup process in general.

See Commander for a description of the various R Commander options.

Author(s)

John Fox

See Also

Commander, Startup.

Scatter3DDialog *Rcmdr 3D Scatterplot Dialog*

Description

Rcmdr 3D Scatterplot Dialog

Usage

```
Scatter3D()
```

Details

Dialog for 3D Scatterplot function.

This dialog sets up a call to the scatter3d function to draw a three-dimensional scatterplot, and optionally to Identify3d to label points interactively with the mouse.

The explanatory variables provide the "horizontal" and "out-of-screen" axes of the scatterplot, the response variable provides the "vertical" axis.

Data points are represented as spheres or points, depending upon the number of observations.

Several regression surfaces can be plotted: a linear least-squares surface; a full quadratic least-squares surface with squared and cross-product terms; a "smooth" regression surface — either a smoothing spline, if no degrees of freedom are specified (in which case the gam function selects the df by generalized cross validation), or a fixed-df regression spline; an additive-regression surface (also fit by gam), with either smoothing spline or regression spline components (again selected according to the specification of degrees of freedom). If only one surface is fit, then residuals are plotted as red (negative) and green (positive) lines from the surface to the points.

You can specify a factor defining groups by pressing the *Plot by groups* button. A separate surface or set of surfaces is plotted for each level of the groups factor. These surfaces can be constrained to be parallel.

The completed plot can be manipulated with the mouse: Click, hold, drag the left mouse button to rotate the display; click, hold, and drag the right button (or centre button on a three-button mouse) to zoom in and out.

If the box labelled *Identify observations with mouse* is checked, you may use the mouse to identify points interactively: Press the right mouse button (or the centre button on a three-button mouse), drag a rectangle around the points to be identified, and release the button. Repeat this procedure for each point or set of "nearby" points to be identified. To exit from point-identification mode, right-click (or centre-click) in an empty region of the plot.

Points may also be identified subsequently by selecting *Identify observations with mouse* from the R Commander *3D graph* menu: As above, click and drag the left mouse button to rotate the display, and click and drag the right (or centre) button to identify points.

Author(s)

John Fox

See Also

scatter3d, Identify3d, rgl-package, gam.

`ScriptEditor` *R Commander Script Editor*

Description

R Commander Script Editor

Details

The R Commander script editor is meant to edit scripts in text widgets, such as the R Commander R Markdown and knitr document tabs.

Saving the document, either via the File menu or pressing the OK button closes the editor and modifies the content of the corresponding R Markdown or knitr tab. Closing the editor without saving, by selecting Cancel from the file menu, pressing the Cancel button or destroying the window

discards changes to the document. You may also save your edits to the R Markdown or knitr tab without closing the editor. Compiling the document into a report also saves the current edits.

The editor is a "non-modal" dialog, and so may remain open when you work.

Author(s)

John Fox

See Also

using R Markdown; using knitr.

Index