Package ‘pmg’

February 20, 2015

Version 0.9-43
Title Poor Man’s GUI
Author John Verzani with contributions by Yvonnick Noel
Maintainer John Verzani <jverzani@gmail.com>
Depends gWidgets, gWidgetsRGtk2, proto, lattice
Imports foreign, MASS

Suggests
Enhances reshape, cairoDevice
Description Simple GUI for R using gWidgets.
License GPL (>= 2)

URL http://www.math.csi.cuny.edu/pmg

R topics documented:

pmg-package .................................................. 2
pmg .............................................................. 2
pmg-dynamic .................................................. 4
pmg-undocumented ........................................ 5
pmgRepeatTrials ............................................. 5

Index 7

1
pmg-package

Poor Man's GUI

Description

Simple GUI for R using RGtk2 and iWidgetsRGtk

Details

Further information is available in the following vignettes:

manual  pmg (source, pdf)

Author(s)

John Verzani  Maintainer: John Verzani <pmgrgui@gmail.com>  URL: http://www.math.csi.cuny.edu/pmg

pmg

A function to start the pmg GUI

Description

The PMG GUI is a simple GUI for R using RGtk2 as the graphical toolkit. The GUI is written using the gWidgets interface to a toolkit.

Usage

pmg(cliType="console", width=1000, height=.75*width, guiToolkit="RGtk2")

pmg.add(widget, label)

pmg.gw(lst, label=NULL)

pmg.addMenubar(menulist)

pmg.eval(command, assignto=NULL)

Arguments

cliType  Where to send output of function called within pmg? This can be either "console" to put output into console that called pmg, or "GUI" to put output into a widget.

width  Width in pixels of initial window
### pmg

**height**  
height in pixels of initial window

**guiToolkit**  
Specify toolkit to use with gWidgets

**widget**  
A gWidgets widget to add to the main notebook for holding dialogs

**label**  
A string containing a label to put on the tab when adding a widget to the main notebook for holding dialogs

**lst**  
A value passed to ggenericwidget. Can be a list, a function name or a function

**menulist**  
A list passed to gmenu for adding to the menubar

**command**  
A string containing a command to be parsed and evaluated in the global environment

**assignto**  
If non-NULL, a variable name to assign the output generated from evaluating the command

### Details

The user can add to the menubar at start up time by defining a list that is called by gmenu. PMG look for a variable pmg.user.menu. This is a list with named components. Each name becomes the menubar entry top level, and each component is called by gmenu to populate the menubar entry.

The functions pmg.add, pmg.gw, pmg.addMenubar, and pmg.eval are used to extend the GUI.

- **pmg.add**: This is used to add a widget to the main notebook containing the dialogs
- **pmg.gw**: This is used to add a ggenericwidget instance to the main notebook containing the dialogs. These widgets can be generated from a function name using the values from formals
- **pmg.addMenubar**: Used to add top-level entries to the main menubar
- **pmg.eval**: Used to send a command, as a string, to the Commands area to be evaluated. Evaluation is done in the global environment.

### Author(s)

John Verzani

### References


### Examples

```r
## Not run:
## this restarts the GUI if the main window has been closed
pmg()

## End(Not run)
```
"Dynamic" widgets for pmg

Description

We call a widget "dynamic" if it updates itself immediately when an event occurs, such as a drag and drop, or a change in some value. The dynamic widgets documented here, are meant to provide quick, easy (but limited) access to R’s modeling functions, R’s significance tests, and R’s lattice functions.

Usage

dModelsDialog()
dTestsDialog()
dLatticeExplorer(container = NULL, ...)

Arguments

container A container to attach the object to
... Currently ignored

Details

For each "dynamic" widget, the variables can be specified by drag and drop, or by editing the widget. The bold-face areas of each widget can be edited by clicking on them or by dropping values. If the drop value comes from a column of an idf instance, then when that column is edited, the dynamic widget is updated. Such variables can not be edited or changed. Other variables may, such as writing powers, or applying functions.

The "dynamic" widgets are meant for easy exploration, but not for saving of actions.

The dModelsDialog shows an interface to lm, aov, and rlm. The user can only specify formulas of the type y ~ 1 + x1 + x2 + ... + xn. Dropping a value on "response" changes the response. Dropping a value on the right side of the ~ adds the term (using +). If the terms are edited by clicking, the values are split on the + sign.

For each model fit, a drop list allows one to generate several of the diagnostic plots.

The dTestsDialog offers an interface to most of the tests in the stats package of class ctests. (The chisq.test is not implemented yet.) Not only can variables be dropped, but one can also change, as appropriate, the choice of the null, the alternative, etc. Again, the bold-face terms may be edited by clicking on them.

The dLatticeExplorer function creates a dynamic graphing widget based on lattice graphics. Up to three variables (only 2 for univariate graphs) may be dropped on the widget. The order is for univariate graphs: ~x then ~x | y. And for bivariate graphs x, x ~ y, x ~ y | z. The panel functions add to the plots of dots by, typically, incorporating some trend line.
Value

Although there are methods for dModelsDialog, these widgets aren’t meant to be interacted with from the command line.

Note

Some of the usability was inspired by the Fathom software.

Author(s)

John Verzani

Examples

```r
# Not run:
dTestsDialog()
# End(Not run)
```

---

### pmg-undocumented

Undocumented, but exported, functions

---

#### pmgRepeatTrials

A function to simplify simulations

---

Description

A simple function to repeat an expression several times as an aid to simplifying simulations.

Usage

```
pmgRepeatTrials(expr, n = 10)
```

Arguments

- **expr** An R expression, such as `rnorm(1)` or `{x <- rnorm(10); t.test(x)$p.value}` that will be repeated `n` times.
- **n** Number of times to repeat the expressions. The default is 10.
Details
This function aids in doing simulations. Rather than explicitly write a for loop or use sapply this function will call sapply on the expression.
A GUI for this appears in pmg under the Simulation tab. The "quick action" will call the function on the results of the simulation.

Value
The output of a sapply call can be a vector, matrix, ... If it is a vector, it is transposed.

Note
This function and GUI was suggested by Daniel Kaplan at useR!2007

Author(s)
John Verzani

Examples
```r
res <- pmgRepeatTrials(rnorm(1))
hist(res)

g = data.frame(
  father = c(78.5, 78.5, 77.5, 76.0, 75.5),
  mother = c(67.0, 68.0, 68.0, 65.5, 62.0),
  sex = c("M", "M", "F", "F", "M"),
  nkids = c(4, 4, 1, 2, 5)
)
res <- pmgRepeatTrials(coef(lm(father~ sex + sample(nkids),data=g)),100)
print(res)
```
Index

*Topic datagen
  pmgRepeatTrials, 5
*Topic interface
  pmg, 2
  pmg-dynamic, 4
  pmg-undocumented, 5
*Topic package
  pmg-package, 2

dLatticeExplorer (pmg-dynamic), 4
dModelsDialog (pmg-dynamic), 4
dTestsDialog (pmg-dynamic), 4

kurtosis (pmg-undocumented), 5

plotecdf (pmg-undocumented), 5
  pmg, 2
  pmg-dynamic, 4
  pmg-package, 2
  pmg-undocumented, 5
  pmg-undocumented.Rd (pmg-undocumented), 5
  pmgRepeatTrials, 5

skewness (pmg-undocumented), 5
summarized.t.test (pmg-undocumented), 5