

# pmg – Poor Man's GUI an R GUI for Introductory Statistics

John Verzani

CUNY/The College of Staten Island

useR!2007

- 1 What is PMG?
  - The basic GUI
- 2 Using PMG for typical tasks in an introductory statistics course
  - Exploratory Data Analysis
  - Summary statistics
  - Why dynamic dialogs?
  - Statistical Inference
  - Linear Modeling
  - Extending pmg
- 3 Closing

- PMG is a graphical user interface (GUI) for R
- It is similar to the Rcmdr interface, only using a more modern toolkit
- It is cross-platform (Windows, mac and linux). (It uses RGtk2.)
- It takes advantage of the drag-and-drop features of RGtk2 for many of its dialogs.
- It is well suited for *introductory statistics* courses

## Starting pmg

The `pmg` package must first be installed. It relies on a few CRAN packages, most importantly the `RGtk2` package. (This is multi-platform, but for some platforms system libraries need to be installed).

A command like

```
install pmg
```

```
> install.packages("pmg", dep=TRUE)
```

Might just work. (A script is available for windows to download the gtk libraries.)

The GUI is started with:

```
Start GUI
```

```
> require(pmg)
```

# The basic GUI

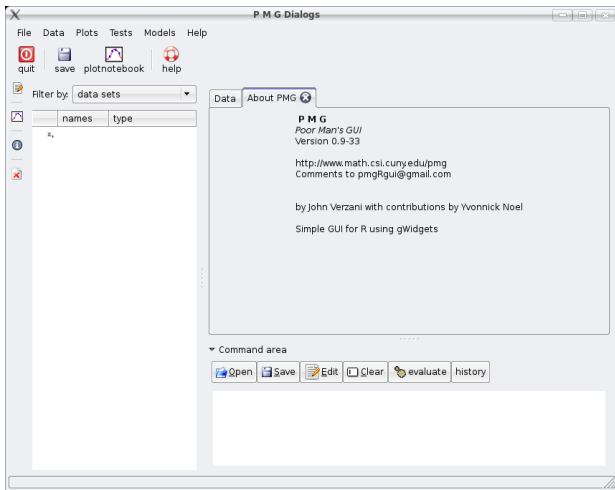


Figure: PMG on startup

# GUI Pieces: The menu bar

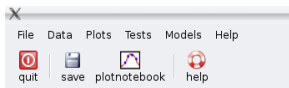


Figure: PMG menubar and toolbar

# The quick drop area



Figure: PMG quick drop area

# A notebook to hold dialogs

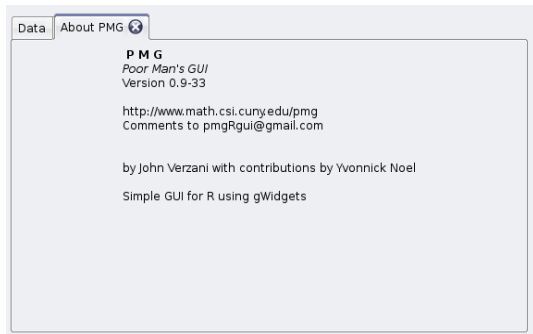


Figure: PMG dialog area



# Command line

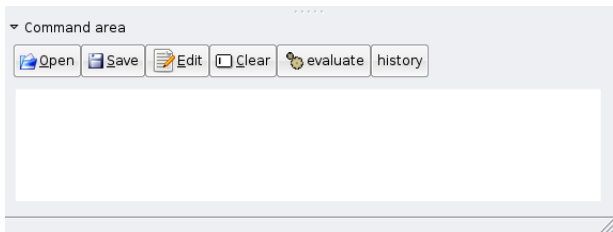


Figure: PMG command area



# Workspace browser area

A data set when loaded appears in the workspace browser area.

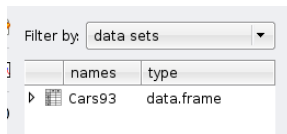


Figure: Data set appearing in workspace browser area

## Other useful dialogs

There are similar dialogs to

- Install a CRAN package (`install.packages`)
- Load a package (`require`)
- Save and restore a workspace
- Source a file
- Browse the help system

# A boxplot

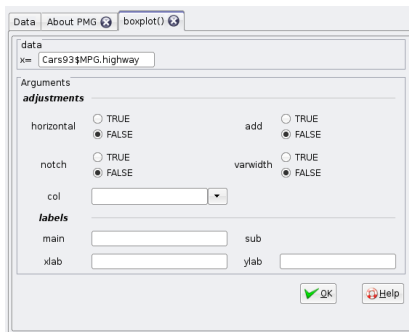
There are atleast two ways to do most basic things in pmg.

- simple dialogs which gather a function's arguments and call the function when "ok" is selected. These print out an R command for students to learn
- A "dynamic dialog" which can be directed easily using just the mouse through drag and drop or clicking, but which does not try to teach the students the R syntax.

One is more instructive and flexible, the other easier to learn.

# The Plots::Univariate::Boxplot dialog

The first way is illustrated with the Plots::Univariate::Boxplot dialog. The figure shows the dialog after some values were filled in.



**Figure:** Basic boxplot dialog for png. There are means to modify each of the boxplot function's arguments.

## Simple dialogs (cont.)

There are a few conveniences with these simple dialogs.

- The variable `Cars93$MPG.highway` was dragged over from the variable browser. This could be typed in. Expressions, such as `rnorm(100)`, are okay, as this value gets evaluated within the global environment.
- The “help” button will call up the help page for the function (`boxplot`), and each label for the arguments, when clicked, will call up the corresponding section from the help page to explain that argument.

# Lattice explorer

The lattice explorer (`Plots::Lattice explorer`) is a “dynamic dialog” and can be used to drag variables from the workspace area to create plots. Dragging and dropping the `MPG.highway` variable, and changing the graphic selector gives the figure.

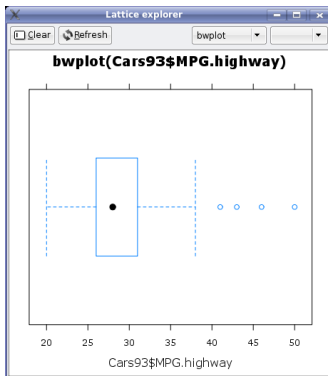
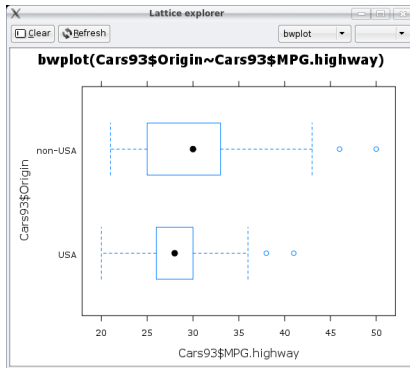


Figure: Boxplot produces using the Lattice Explorer



## Lattice explorer: multivariate

The lattice explorer makes multi-variate explorations easy. Dropping a factor onto the graphic will produce the following figure.



**Figure:** Lattice explorer after a factor is dropped following a numeric variable.

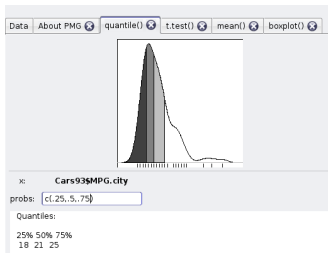
## Lattice explorer (cont.)

Changing the plot selector will produce a different graphic. Use clear to start over with new variables.

Unlike the other dialog, there is no way to change things such as the title or orientation, etc. Again, the trade-off between easier to learn versus flexibility.

# Quantiles

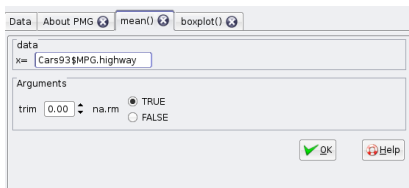
The basic dialogs are more or less generated from the underlying R function's arguments. A few have been optimized such as the quantile dialog.



**Figure:** The `Data::Quantiles` dialog showing a graphic in addition to a summary.

# The mean

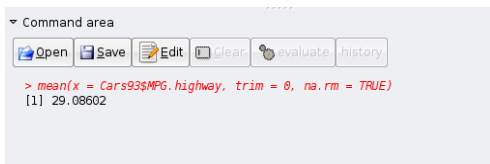
Similarly there are dialogs to compute summary statistics, such as the mean. There is a basic dialog where any applicable arguments can be entered (blanks are left out)



**Figure:** basic dialog for the mean where a few arguments can be adjusted.

cont.

The output of the R command appears in the command area:



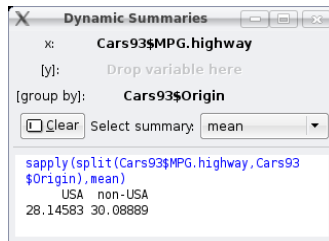
The screenshot shows a window titled "Command area" with a toolbar containing buttons for "Open", "Save", "Edit", "Clear", "evaluate", and "history". Below the toolbar, the R command `> mean(x = Cars93$MPG.highway, trim = 0, na.rm = TRUE)` is entered, and the output `[1] 29.08602` is displayed.

**Figure:** Command area showing output from finding the mean

Unlike the next dynamic dialog, these dialogs produce an R command that can be copied and pasted into a word processor for report writing or, if desired, edited in the commands area.

## the mean using the Dynamic summaries feature

As with the Lattice explorer, there is a more interactive dialog for finding the mean under `Data::Dynamic summaries`. We dragged two variables over to produce the following:

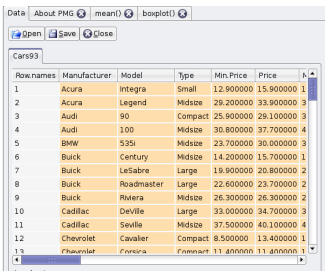


**Figure:** Dynamic summaries dialog showing how to find the mean for grouped data.

# Why dynamic

Some of the design of `pmg` was influenced by the Fathom Dynamic Data software of Key Curriculum Press. There the graphics are tightly linked with a data frame viewer. The data frame viewer in `pmg` can be used the same way. First, drag the data set over the `Data` tab and onto the `Open` button. This will open it in the data frame editor.

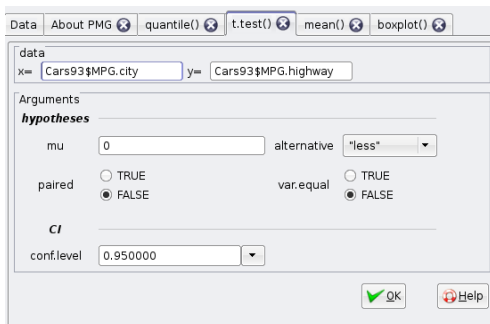
Then drag a column header onto a drop area for a dynamic dialog; and then click open the `subset=` area and select a variable to adjust. The dialog should adjust accordingly.



Row.names	Manufacturer	Model	Type	Min.Price	Price	N
1	Acura	Integra	Small	12.900000	15.900000	1
2	Acura	Legend	Midsize	29.200000	33.900000	3
3	Audi	90	Compact	25.900000	29.100000	3
4	Audi	100	Midsize	30.800000	37.700000	4
5	BMW	535i	Midsize	23.700000	30.000000	3
6	Buick	Century	Midsize	14.200000	15.700000	1
7	Buick	LeSabre	Large	19.900000	20.800000	2
8	Buick	Roadmaster	Large	22.600000	23.700000	2
9	Buick	Riviera	Midsize	26.300000	26.300000	2
10	Cadillac	DeVille	Large	33.000000	34.700000	3
11	Cadillac	Seville	Midsize	37.500000	40.100000	4
12	Chevrolet	Cavalier	Compact	8.500000	13.400000	1
13	Chrysler	Conquest	Compact	11.400000	11.400000	1

# The $t$ test

Statistical inference has similar functionality. There is a  $t$ -test dialog that can be used. In this case we show the bivariate  $t$ -test



The image shows a dialog box titled "t.test()" with several tabs at the top: "Data", "About PMG", "quantile()", "t.test()", "mean()", and "boxplot()". The "Data" tab is selected. Below the tabs, there are two input fields: "x=" with the value "Cars93\$MPG.city" and "y=" with the value "Cars93\$MPG.highway".

The dialog is divided into sections:

- Arguments**
  - hypotheses**
    - mu: 0
    - alternative: "less" (dropdown menu)
  - paired:  TRUE,  FALSE
  - var.equal:  TRUE,  FALSE
- CI**
  - conf.level: 0.950000 (dropdown menu)

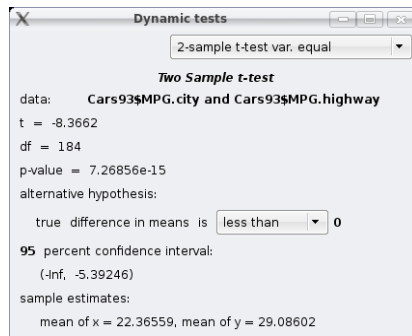
At the bottom right, there are two buttons: "OK" (with a green checkmark) and "Help" (with a red question mark).

Figure: A two-sample  $t$ -test



## Dynamic tests

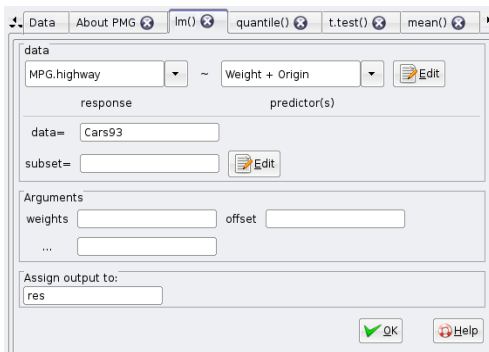
The `Tests::Dynamic tests` dialog takes its output from that of the `htest` class. It should look more or less familiar. The bold text, when clicked, allows for editing of those parameter values. One can use a formula interface by selecting from the top popup widget.



**Figure:** Dynamic tests dialog showing a two-sample  $t$ -test. The bold areas allow for editing.

# Linear modeling

Another common topic in an introductory statistics course is the simple linear model. Again there are two different ways to do this.



**Figure:** Simple dialog for `lm`. The formula was added using the dialog for editing formulas.

# Simple 1m dialog cont.

The model formula editing dialog was modeled on one from S-Plus

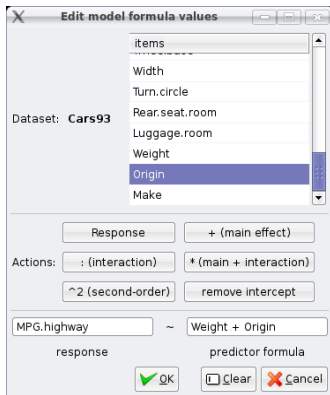
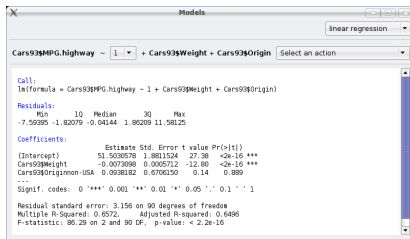


Figure: Model formula editing dialog

# The dynamic models dialog

The dynamic models dialog for linear regression is intended to be used in a drag-and-drop manner. The model formula are built up in a simple manner. (They can be edited if desired). Some quick actions are available from a popup-box on the right.



**Figure:** The dynamic models dialog. Dropping variables produces a simple model formula without interactions

# Extending the GUI

One of the neat parts of Rcmdr has been the interest in extending the GUI (RcmdrPlugin.TeachingDemos, Rcmdr.HH, TsCmdr, ...) for specific purposes.

Currently, no plug-in architecture is supported, but there is a means to add to the toolbar using `pmg.addMenubar`.

Additionally, `pmg` is written using the `gWidgets` package which makes it relatively easy to develop new GUIs. Yvonnick Noel contributed several that are available under the `Plots::TeachingDemos` dialog.

# Teaching Demos Dialog

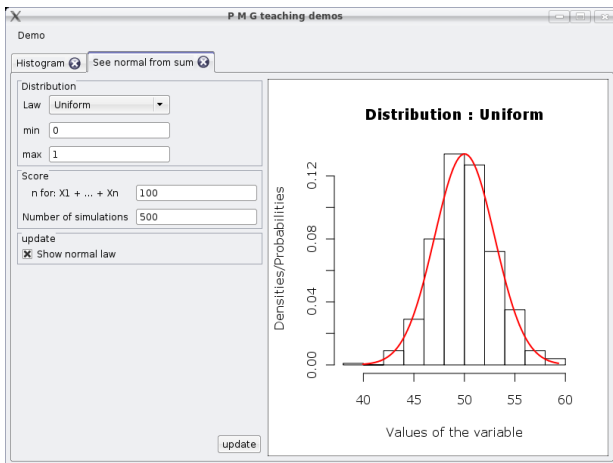


Figure: Teaching demos notebook

# What is lacking?

Hopefully you've been convinced that `pmg` offers an easy-to-learn GUI for R that can easily handle most tasks of an introductory statistics course. Of course, `pmg` could be improved. Here are some immediate areas:

- There is no report writing functionality
- I like the dynamic dialogs, the others have a “website” feel. Are either the right metaphor for introductory students? The “inference for office” project of Josh van Eikeren provides an alternative akin to Mathematica worksheets. (Windows only)
- Rcmdr has a more advanced set of dialogs.
- Rcmdr has spawned a number of topic specific spinoffs (e.g., TsCmdr). Easily adding new menu items could be improved.