

Getting Started With the R Commander: A Basic-Statistics Graphical User Interface to R

John Fox*
McMaster University
Hamilton, Ontario, Canada
21 June 2004

Abstract

Unlike S-PLUS, R does not include a statistical graphical user interface (GUI), but it does include tools for building GUIs. Based on the `tcltk` package (which furnishes an interface to the Tcl/Tk GUI builder) the `Rcmdr` package provides a basic-statistics graphical user interface to R called the “R Commander.”

The design objectives of the R Commander were as follows: to support, through an easy-to-use, extensible, cross-platform GUI, the statistical functionality required for a basic-statistics course (though its current functionality has grown to include support for linear and generalized-linear models); to make it relatively difficult to do unreasonable things; and to render visible the relationship between choices made in the GUI and the R commands that they generate.

The R Commander uses a simple and familiar menu/dialog-box interface. Top-level menus include *File*, *Edit*, *Data*, *Statistics*, *Graphs*, *Models*, *Distributions*, and *Help*, with the complete menu tree given in the paper. Each dialog box includes a *Help* button, which leads to a relevant help page.

Menu and dialog-box selections generate R commands, which are recorded in a log/script window and are echoed, along with output, to an output window. The log/script window also provides the ability to edit, enter, and re-execute commands.

Data sets in the R Commander are simply R data frames, and can be read from attached packages or imported from files. Although several data frames may reside in memory, only one is “active” at any given time.

The purpose of this paper is to introduce and describe the basic use of the R Commander GUI and the manner in which it can be extended. Most of the paper can serve as an introductory guide for students who will use the R Commander

*Please address correspondence to jfox@mcmaster.ca. The work described in this paper was supported by a grant from the Social Science Research Board of McMaster University. Many individuals – too numerous to name here – have made helpful suggestions for the `Rcmdr` package; please see the `Changes` file distributed with the package for their names. I am also grateful to Tony Christensen for research assistance and to Bob Andersen for comments on a draft of this paper. This is a revised version of a paper presented at the useR! Conference, Vienna, May 2004.

Background

R (Ihaka and Gentleman, 1996; R Development Core Team, 2004) is a free, open-source implementation of the S statistical computing language and programming environment. R is a command-driven system: One normally specifies a statistical analysis in R by typing commands – that is, statements in the S language that are executed by the R interpreter. S-PLUS (a commercial implementation of the S language), also provides a graphical user interface (a “GUI”) to much of the statistical functionality of S.

In my opinion, a GUI for statistical software is a mixed blessing: On the one hand, a GUI doesn’t require that the user remember the names and arguments of commands, and decreases the chances of syntax and typing errors. These characteristics make GUIs particularly attractive for introductory, casual, or infrequent use of software. On the other hand, having to drill one’s way through successive layers of menus and dialog boxes can be tedious and can make it difficult to reproduce a statistical analysis, perhaps with variations. Moreover, providing a GUI for a statistical system that includes hundreds (or even thousands) of commands, many incorporating extensive options, can produce a labyrinth.

Unlike S-PLUS, R does not include a statistical GUI, but it does furnish tools for building GUIs.¹ The `Rcmdr` package provides a basic-statistics GUI for R, which I call the “R Commander.” The design objectives of the R Commander were as follows:

- Most important, to provide, through an easy-to-use, cross-platform², extensible GUI, the statistical functionality required for a basic-statistics course. The original target text was David Moore’s *The Basic Practice of Statistics, Second Edition* (Freeman, 2000). With the help of a research assistant (Tony Christensen), I’ve since examined several other texts (including the third edition of Moore, 2004), collected suggestions from a number of individuals, and slightly expanded the horizons of the R Commander – for example, to include linear and generalized-linear models.
- To make it relatively difficult to do unreasonable things (such as specifying a categorical variable as the response in a linear regression).
- To render visible the relationship between choices made in the GUI and the R commands that they generate. Commands are both pasted into a log/script window in the *R Commander* and echoed to an output window (see below). The

¹ The R Commander, described in this paper, is based on the `tcltk` package (Dalgaard, 2001, 2002), which provides an interface to Tcl/Tk.

² The examples in this document use the Windows version of R, and parts of the document are specific to the Windows version. R, however, is available on other computing platforms as well (Macintosh computers and Unix/Linux systems), and the use of R and the R Commander on these other systems is very similar to their use under Windows. I focus here on the Windows version of the software because I believe that the large majority of students in basic-statistics classes are Windows users.

log/script window is editable, commands in the window can be executed or re-executed, and new commands can be entered. Scripts can also be saved to and loaded from files.

The principal purpose of this paper is to introduce and describe the basic use of the R Commander GUI. In particular, most of the paper can serve as an introductory guide for students who will use the R Commander. In addition, the penultimate section of the paper explains how the R Commander can be extended, and the final section provides some information for instructors. The help files for the `Rcmdr` package appear as an appendix to the paper.

Starting R and the R Commander

Once R has started up, simply loading the `Rcmdr` package by typing the command `library(Rcmdr)` into the *R Console*, starts the R Commander GUI. To function properly under Windows, the R Commander requires the single-document interface (SDI) to R.³ After loading the package, *R Console* and *R Commander* windows should appear more or less as in Figure 1. Figure 1 and other screen images in this document were created under Windows XP configured to “classic” appearance; if you use another version of Windows (or, of course, another computing platform), then the appearance of the screen may differ.

³ The Windows version of R is normally run from a multiple-document interface (“MDI”), which contains the *Console* window and *Graphical Device* windows created during the session. In contrast, under the single-document interface (“SDI”), the *R Console* and *Graphical Device* windows are not contained within a master window. There are several ways to run R in SDI mode – for example, by editing the `Rconsole` file in R’s `etc` subdirectory, or by adding `--sdi` to the *Target* field in the *Shortcut* tab of the R desktop icon’s *Properties*. This limitation of the `Rcmdr` package is inherited from the `tcltk` package, on which `Rcmdr` depends.

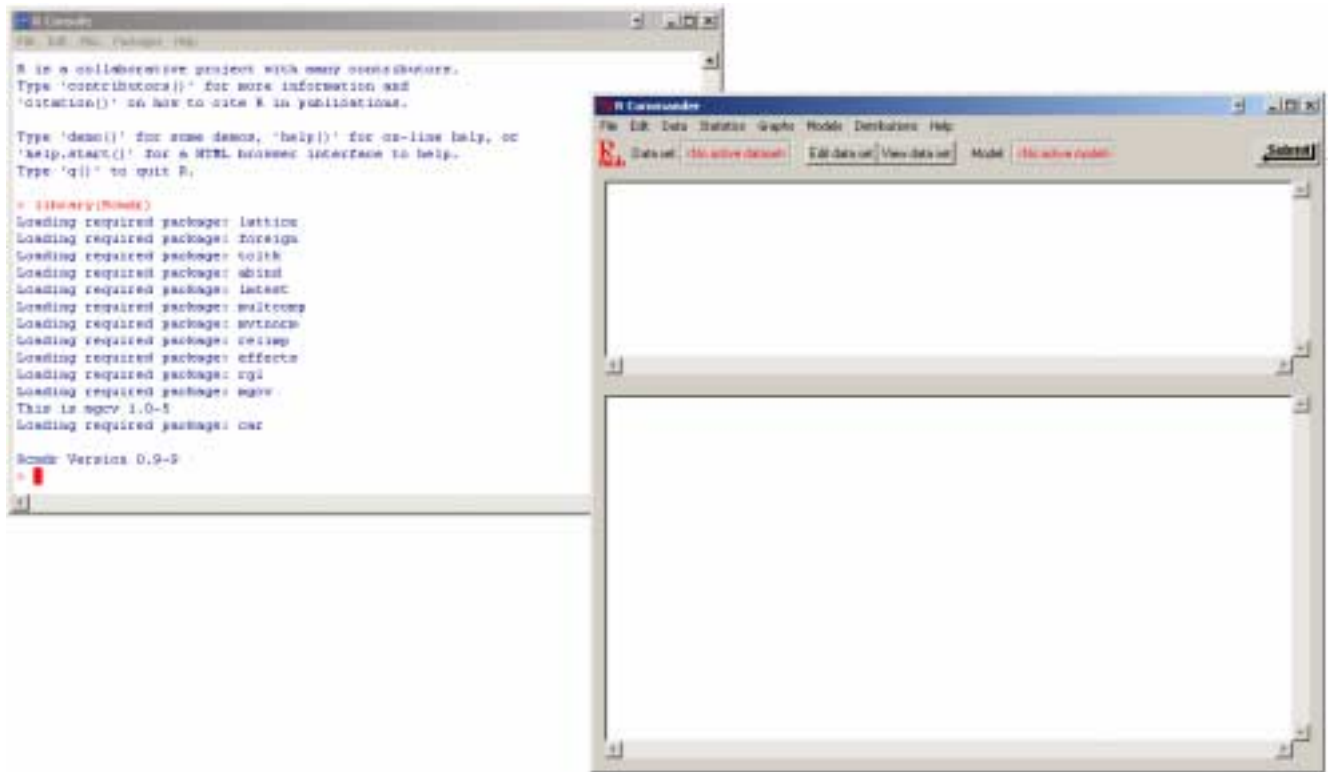


Figure 1: The R Console and R Commander windows at start-up.

The *R Commander* and *R Console* windows float freely on the desktop.⁴ You will normally use the menus and dialog boxes of the *R Commander* to read, manipulate, and analyze data. Printed output appears by default in the lower text window in the *R Commander* window. When you create graphs, these will appear in a separate *Graphics Device* window. You can also type R commands at the `>` (greater-than) prompt in the *R Console* or into the upper text window in the *R Commander*; the main purpose of the *R Commander*, however, is to avoid typing commands.

Notice the menus along the top of the *R Commander* window:

- *File*: Menu items for loading and saving log files; setting options; and exiting.
- *Edit*: Menu items (*Cut*, *Copy*, *Paste*, etc.) for editing the contents of the log/script and output windows. Right clicking in the script or output window also brings up an edit “context” menu.

⁴ Notice that *Rcmdr* requires some packages in addition to several of the “recommended” packages that are normally distributed with R, and loads these packages at startup. *Rcmdr*, the required packages, and many other contributed packages are available for download from the Comprehensive R Archive Network (CRAN) at <http://cran.r-project.org/>.

- *Data*: Submenus containing menu items for reading and manipulating data.
- *Statistics*: Submenus containing menu items for a variety of basic statistical analyses.
- *Graphs*: Menu items for creating simple statistical graphs.
- *Models*: Menu items for obtaining numerical summaries and graphs for a linear or generalized linear model, and for adding diagnostic quantities, such as residuals, to the data set.
- *Distributions*: Probabilities, quantiles, and graphs of standard statistical distributions (to be used, for example, as a substitute for statistical tables).
- *Help*: Menu items to obtain information about the R Commander (including this paper). As well, each R Commander dialog box has a *Help* button (see below).

The complete menu “tree” for the R Commander (version 0.9-9) is shown in Figure 2. Most menu items lead to dialog boxes, as illustrated later in this document.

```
File - Load log from file
      |- Save log
      |- Save log as
      |- Options
      |- Reset output width
      |- Exit - from Commander
          |- from Commander and R

Edit - Clear window
      |- Cut
      |- Copy
      |- Paste
      |- Delete
      |- Find
      |- Select all

Data - New data set
      |- Import data - from text file
      |               |- from SPSS data set
      |               |- from Minitab data set
      |- Data in packages - List data sets in packages
      |               |- Read data set from attached package
      |- Active data set - Select active data set
      |               |- Help on active data set (if available)
      |               |- Variables in active data set
      |               |- Set case names
      |               |- Subset active data set
      |               |- Remove cases with missing data
      |               |- Export active data set
      |- Manage variables in active data set - Recode variable
                                              |- Compute new variable
                                              |- Standardize variables
                                              |- Convert numeric variable to factor
                                              |- Reorder factor levels
                                              |- Define contrasts for a factor
                                              |- Rename variables
                                              |- Delete variables from data set
```

```

Statistics - Summaries - Active data set
|
|   |- Numerical summaries
|   |- Frequency distribution
|   |- Table of statistics
|   |- Correlation matrix
|- Contingency Tables - Two-way table
|
|   |- Multi-way table
|   |- Enter and analyze two-way table
|- Means - Single-sample t-test
|
|   |- Independent-samples t-test
|   |- Paired t-test
|   |- One-way ANOVA
|   |- Multi-way ANOVA
|- Proportions - Single-sample proportion test
|
|   |- Two-sample proportions test
|- Variances - Two-variances F-test
|
|   |- Bartlett's test
|   |- Levene's test
|- Nonparametric tests - Two-sample Wilcoxon test
|
|   |- Paired-samples Wilcoxon test
|   |- Kruskal-Wallis test
|- Dimensional analysis - Scale reliability
|
|   |- Principal-components analysis
|   |- Factor analysis
|- Fit models - Linear regression
|
|   |- Linear model
|   |- Generalized linear model

Graphs - Index plot
|- Histogram
|- Stem-and-leaf display
|- Boxplot
|- Quantile-comparison plot
|- Scatterplot
|- Scatterplot matrix
|- 3D scatterplot
|- Line graph
|- Plot of means
|- Bar graph
|- Pie chart

```

```

Models - Select active model
| - Summarize model
| - Add observation statistics to data
| - Hypothesis tests - ANOVA table
|   | - Compare two models
|   | - Linear hypothesis
| - Numerical diagnostics - Variance-inflation factors
|   | - Breusch-Pagan test for heteroscedasticity
|   | - Durbin-Watson test for autocorrelation
|   | - RESET test for nonlinearity
|   | - Bonferroni outlier test
| - Graphs - Basic diagnostic plots
|   | - Residual quantile-comparison plot
|   | - Component+residual plots
|   | - Added-variable plots
|   | - Influence plot
|   | - Effect plots

Distributions - Normal distribution - Normal quantiles
|   | - Normal probabilities
|   | - Plot normal distribution
| - t distribution - t quantiles
|   | - t probabilities
|   | - Plot t distribution
| - Chi-squared distribution - Chi-squared quantiles
|   | - Chi-squared probabilities
|   | - Plot chi-squared distribution
| - F distribution - F quantiles
|   | - F probabilities
|   | - Plot F distribution
| - Binomial distribution - Binomial quantiles
|   | - Binomial tail probabilities
|   | - Binomial probabilities
|   | - Plot binomial distribution
| - Poisson distribution - Poisson probabilities
|   | - Plot Poisson distribution

Help - Commander help
| - About Rcmdr
| - Introduction to the R Commander

```

Figure 2: The R-Commander menu tree.

The R Commander interface includes a few elements in addition to the menus and dialogs:

- Below the menus is “toolbar” with a row of buttons.
 - The leftmost (flat) button shows the name of the active data set. Initially there is no active data set. If you press this button, you will be able to choose among data sets currently in memory. Most of the menus and dialogs in the R Commander reference the active data set. (The *Distributions* and *Models* menus are exceptions.)
 - Two buttons allow you to open a data editor to modify the active data set or a viewer to examine it. The data-set viewer can remain open while other operations are performed.
 - A flat button indicates the name of the active statistical model – either a linear model (such as a linear-regression model) or a generalized linear model. Initially there is no active model. If there is more than one model in memory, you can choose among them by pressing the button.
 - A *Submit* button, at the far right of the toolbar, allows you to execute selected commands from the script window.
- Immediately below the toolbar is the log/script window, a large scrollable text window. As mentioned, commands generated by the GUI are normally copied into this window. You can edit the text in the log/script window or even type your own R commands into the window. Pressing the *Submit* button (or, alternatively, the key combination *Ctrl-s*) causes the line containing the cursor to be submitted (or resubmitted) for execution. If several lines are selected (e.g., by left-clicking and dragging the mouse over them), then pressing *Submit* will cause all of them to be executed. Commands entered into the log-script window can extend over more than one line, but if they do, lines after the first must be indented with one or more spaces or tabs.
- At the bottom is a large scrollable and editable text window for output. Commands echoed to this window appear in red, output in blue.

Once you have loaded the `Rcmdr` package, you can minimize the *R Console*. The *R Commander* window can also be resized or maximized in the normal manner. If you make the *R Commander* window wider, then you may wish to reset the width of printed output from R via the *File* → *Reset output width...* menu.⁵

The R Commander is highly configurable: I have described the default configuration here. Changes to the configuration can be made via the *File* → *Options...* menu, or by setting options in R. See the `Rcmdr` help files for details.

⁵ A menu item that terminates in ellipses (i.e., the dots, ...) leads to a dialog box; this is a standard GUI convention. . In this document, → represents selecting a menu item or submenu from a menu.

Data Input

Most of the procedures in the R Commander assume that there is an *active data set*.⁶ If there are several data sets in memory, you can choose among them, but only one is active. When the R Commander starts up, there is no active data set.

The R Commander provides several ways to get data into R:

- You can enter data directly via *Data* → *New data set*.... This is a reasonable choice for a very small data set.
- You can import data from a plain-text (“ascii”) file or from another statistical package (Minitab or SPSS).
- You can read a data set that is included in an R package, either typing the name of the data set (if you know it), or selecting the data set in a dialog box.

Reading Data From a Text File

For example, consider the data file `Nations.txt`.⁷ The first few lines of the file are shown in Figure 3:

- The first line of the file contains variable names: `TFR` (the total fertility rate, expressed as number of children per woman), `contraception` (the rate of contraceptive use among married women, in percent), `infant.mortality` (the infant-mortality rate per 1000), `GDP` (gross domestic product per capita, in U.S. dollars), and `region`.
- Subsequent lines contain the data values themselves, one line per country. The data values are separated by “white space” – one or more blanks or tabs. Although it is helpful to make the data values line up vertically, it is not necessary to do so. Notice that the data lines begin with the country names. Because we want these to be the “row names” for the data set, there is no corresponding variable name: That is, there are five variable names but six data values on each line. When this happens, R will interpret the first value on each line as the row name.
- Some of the data values are missing. In R, it is most convenient to use `NA` (representing “not available”) to encode missing data, as I have done here.
- The variables `TFR`, `contraception`, `infant.mortality`, and `GDP` are numeric (quantitative) variables; in contrast, `region` contains region names.

⁶ Procedures selected under via the *Distributions* menu are exceptions, as is *Enter and analyze two-way table* under the *Statistics* → *Contingency tables* menu.

⁷ This file is included in the `etc` subdirectory of the `Rcmdr` package.

When the data are read, R will treat `region` as a “factor” – that is, as a categorical variable. In most contexts, the R Commander distinguishes between numerical variables and factors.

	TFR	contraception	infant.mortality	GDP	region	
Afghanistan	6.90	NA	154	2848	Asia	
Albania	2.60	NA	32	863	Europe	
Algeria	3.81	52	44	1531	Africa	
American-Samoa	NA	NA	11	NA	Oceania	
Andorra	NA	NA	NA	NA	Europe	
Angola	6.69	NA	124	355	Africa	
Antigua	NA	53	24	6966	Americas	
Argentina	2.62	NA	22	8055	Americas	
Armenia	1.70	22	25	354	Europe	
Australia	1.89	76	6	20046	Oceania	
.						
.						
.						

Figure 3: The first few lines of the data file `Nations.txt`.

To read the data file into R, select *Data* → *Import data* → *from text file ...* from the R Commander menus. This operation brings up a *Read Data From Text File* dialog, as shown in Figure 4. The default name of the data set is `Dataset`. I’ve changed the name to `Nations`.

Valid R names begin with an upper- or lower-case letter (or a period, `.`) and consist entirely of letters, periods, underscores (`_`), and numerals (i.e., 0–9); in particular, do not include any embedded blanks in a data-set name. You should also know that R is case-sensitive, and so, for example, `nations`, `Nations`, and `NATIONS` are distinguished, and could be used to represent different data sets.

Clicking the *OK* button in the *Read Data From Text File* dialog brings up an *Open* file dialog, also shown in Figure 4. Here I navigated to the file `Nations.txt`. Clicking the *Open* button in the *Open* file dialog will cause the data file to be read. Once the data file is read, it becomes the active data set in the R Commander. As a consequence, in Figure 5, the name of the data set appears in the data set button near the top left of the R Commander window.

I clicked the *View data set* button to bring up the data viewer window (from David Firth’s `relimp` package) also shown in Figure 5. Notice that the commands to read and attach the `Nations` data set (the R `read.table` and `attach` commands) appear, partially obscured by the display of the data set, in log/script and output windows.

The `read.table` command creates an R data frame, which is a representation of a rectangular cases-by-variables data set: The rows of the data set represent cases or observations and the columns represent variables. Data sets in the R Commander are R data frames.

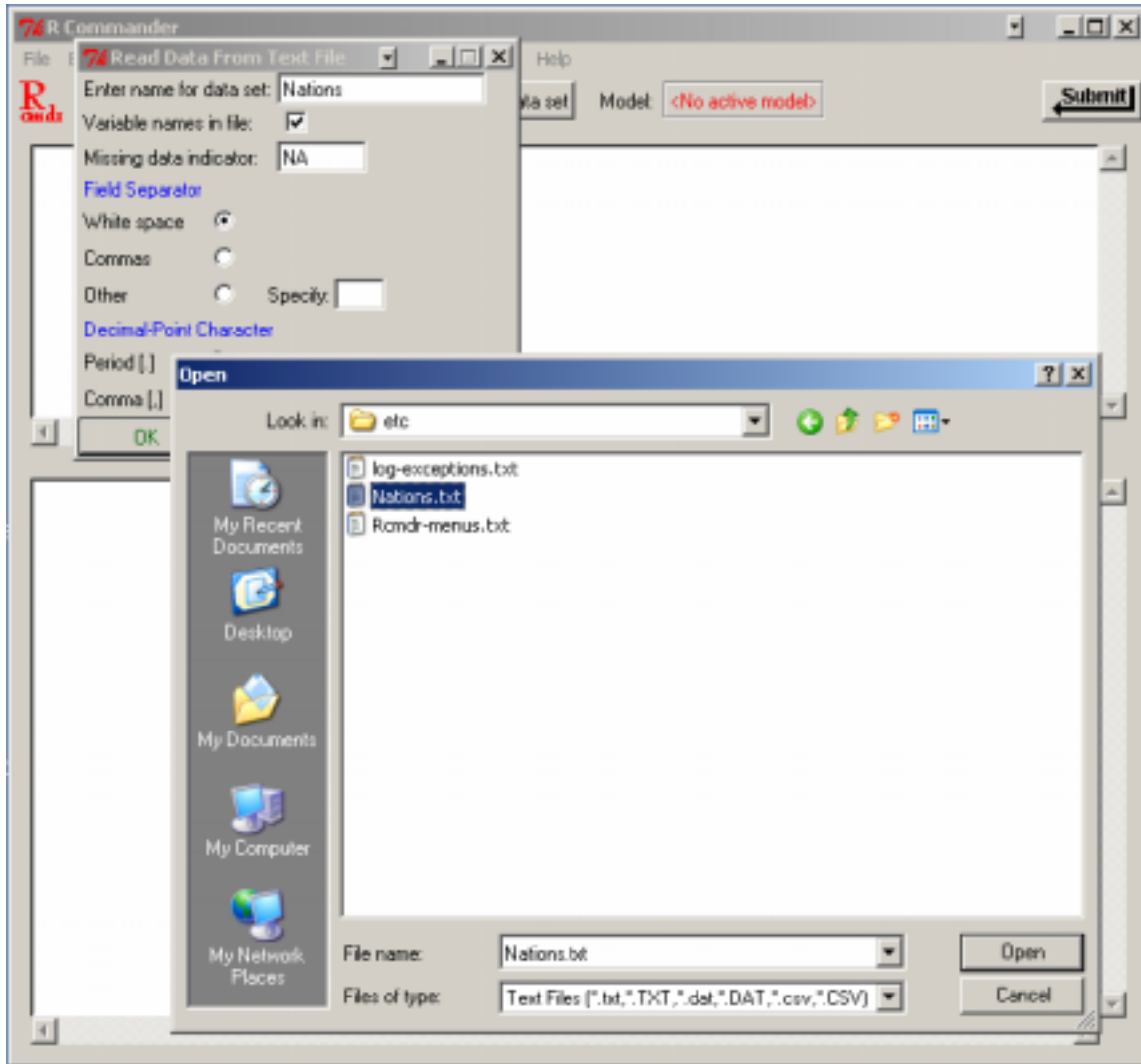


Figure 4: Reading data from a text file.

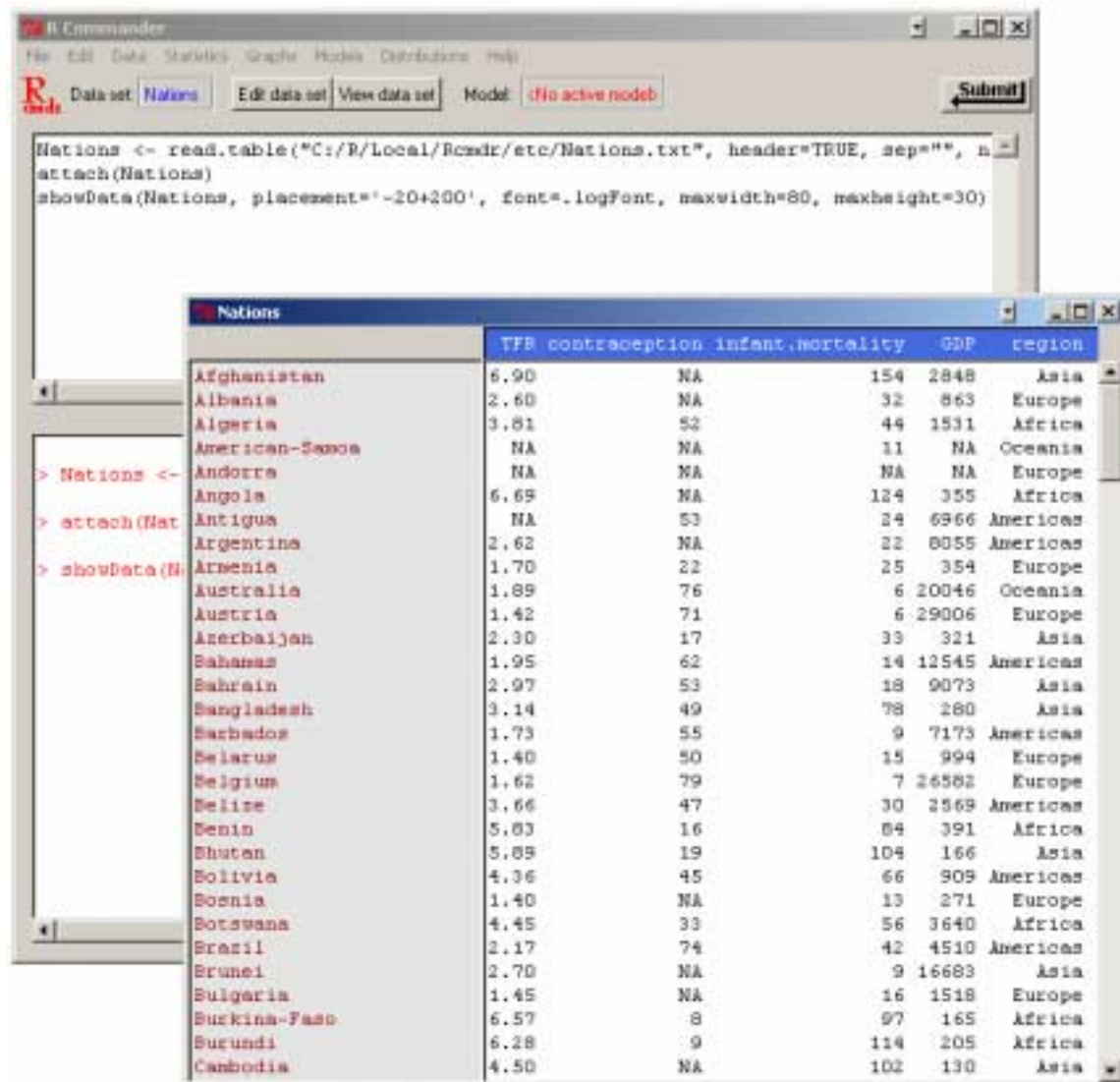


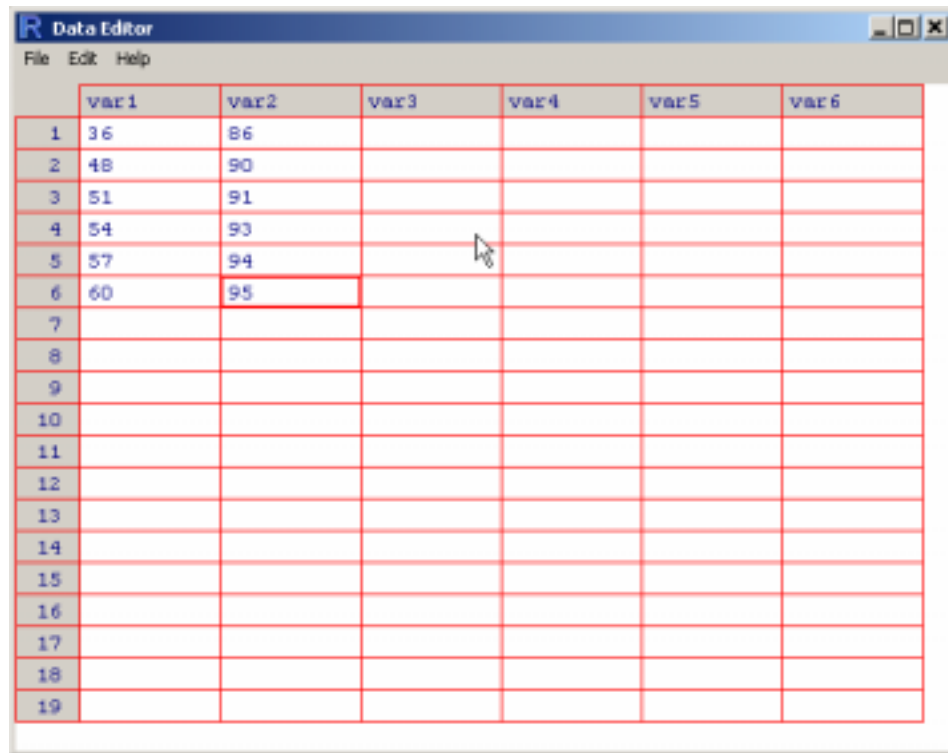
Figure 5: Displaying the active data set.

Entering Data Directly

To enter data directly into the spreadsheet data editor you can proceed as follows. As an example, I use a very small data set from Problem 2.44 in Moore (2000):

- *Select Data* → *New data set...* from the *R Commander* menus. Optionally enter a name for the data set, such as `Problem2.44`, in the resulting dialog box, and click the *OK* button. (Remember that R names cannot include intervening blanks.) This will bring up a *Data Editor* window with an empty data set.

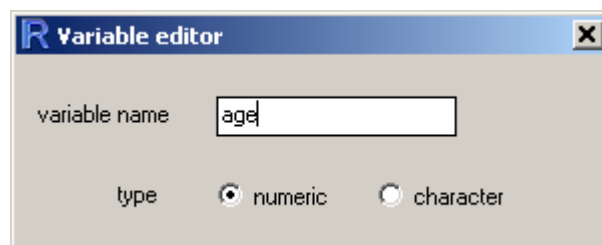
- Enter the data from the problem into the first two columns of the data editor. You can move from one cell to another by using the arrow keys on your keyboard, by tabbing, by pressing the *Enter* key, or by pointing with the mouse and left-clicking. When you are finished entering the data, the window should look like Figure 6.



	var1	var2	var3	var4	var5	var6
1	36	86				
2	48	90				
3	51	91				
4	54	93				
5	57	94				
6	60	95				
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						

Figure 6: Data editor after the data are entered.

- Next, click on the name `var1` above the first column. This will bring up a *Variable editor* dialog box, as in Figure 7.



R Variable editor

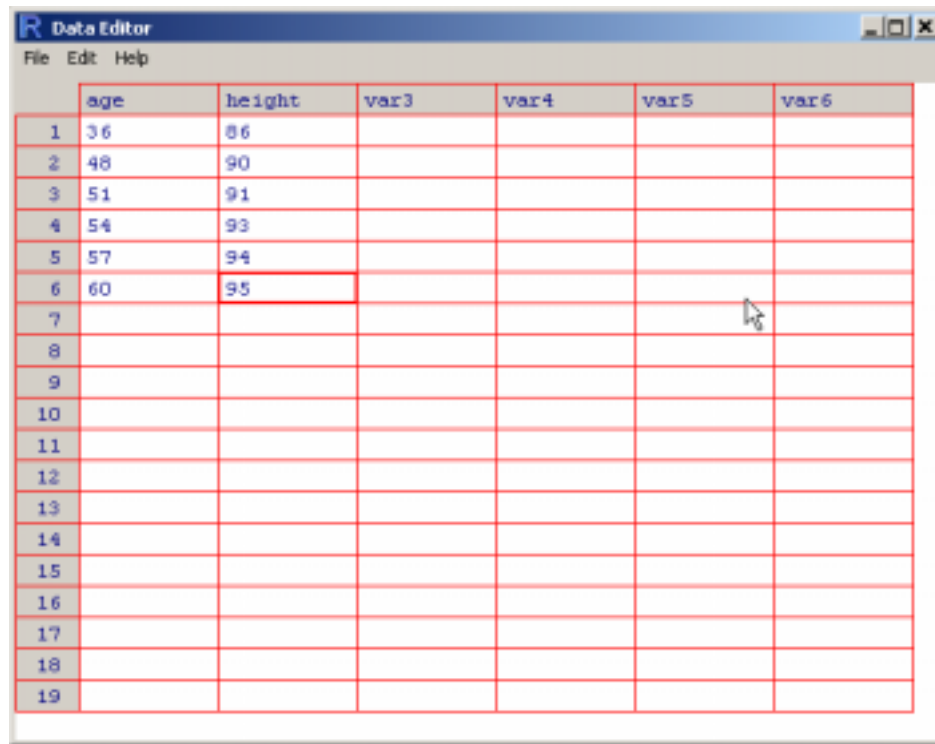
variable name

type ☒ numeric ☐ character

Figure 7: Dialog box for changing the name of a variable in the data editor.


- Type the variable name `age` in the box, just as I have, and click the ☒ button in the upper-right corner of the *Variable editor* window, or press the *Enter* key, to close the window. Repeat this procedure to name the second column `height`.

The *Data Editor* window should now look like Figure 8.



	age	height	var3	var4	var5	var6
1	36	86				
2	48	90				
3	51	91				
4	54	93				
5	57	94				
6	60	95				
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						

Figure 8: The data editor window after box variable names have been changed.

- Select *File* → *Close* from the *Data Editor* menus or click the  at the upper-right of the *Data Editor* window. The data set that you entered is now the active data set in the *R Commander*.

Creating Numerical Summaries and Graphs

Once there is an active data set, you can use the *R Commander* menus to produce a variety of numerical summaries and graphs. I will describe just a few basic examples here. A good GUI should be largely self-explanatory: I hope that once you see how the *R Commander* works, you will have little trouble using it, assisted perhaps by the on-line help files.

In the examples below, I assume that the active data set is the `Nations` data set, read from a text file in the previous section. If you typed in the five-observation data set from Moore (2000) also described in the previous section, then that is the active data set. Recall that you can change the active data set by clicking on the flat button with the active data set's name near the top of the *R Commander* window, selecting from among a list of data sets currently resident in memory.

Selecting *Statistics* → *Summaries* → *Active data set* produces the results shown in Figure 9. For each numerical variable in the data set (TFR, contraception, infant.mortality, and GDP), R reports the minimum and maximum values, the first and third quartiles, the median, and the mean, along with the number of missing values. For the categorical variable *region*, we get the number of observations at each “level” of the factor. Had the data set included more than ten variables, the R Commander would have asked us whether we really want to proceed – potentially protecting us from producing unwanted voluminous output.

Similarly, selecting *Statistics* → *Summaries* → *Numerical summaries. . .* brings up the dialog box shown in Figure 10. Only numerical variables are shown in the variable list in this dialog; the factor *region* is missing, because it is not sensible to compute numerical summaries for a factor. Clicking on *infant.mortality*, and then clicking *OK*, produces the following output:

```
> mean(Nations$infant.mortality, na.rm=TRUE)
[1] 43.47761

> sd(Nations$infant.mortality, na.rm=TRUE)
[1] 38.75604

> quantile(Nations$infant.mortality, c( 0,.25,.5,.75,1 ),
           na.rm=TRUE)
 0%   25%   50%   75%  100%
 2    12    30    66   169
```

By default, the R commands that are executed print out the mean and standard deviation of the variables, along with quantiles (percentiles) corresponding to the minimum, the first quartile, the median, the third quartile, and the maximum.

As is typical of R Commander dialogs, the *Numerical Summaries* dialog box in Figure 10 includes *OK*, *Cancel*, and *Help* buttons. The *Help* button leads to a help page either for the dialog itself or (as here) for an R function that the dialog invokes.

Making graphs with the R Commander is also straightforward. For example, selecting *Graphs* → *Histogram* from the *R Commander* menus brings up the *Histogram* dialog box in Figure 11, and clicking on *infant.mortality* followed by *OK*, opens up a *Graphics Device* window with the histogram shown in Figure 12.

If you make several graphs in a session, then only the most recent normally appears in the *Graphics Device* window. You can recall previous graphs using the *Page Up* and *Page Down* keys on your keyboard.⁸

⁸ At startup, the R Commander turns on the graph history mechanism; this feature is available only in Windows systems.

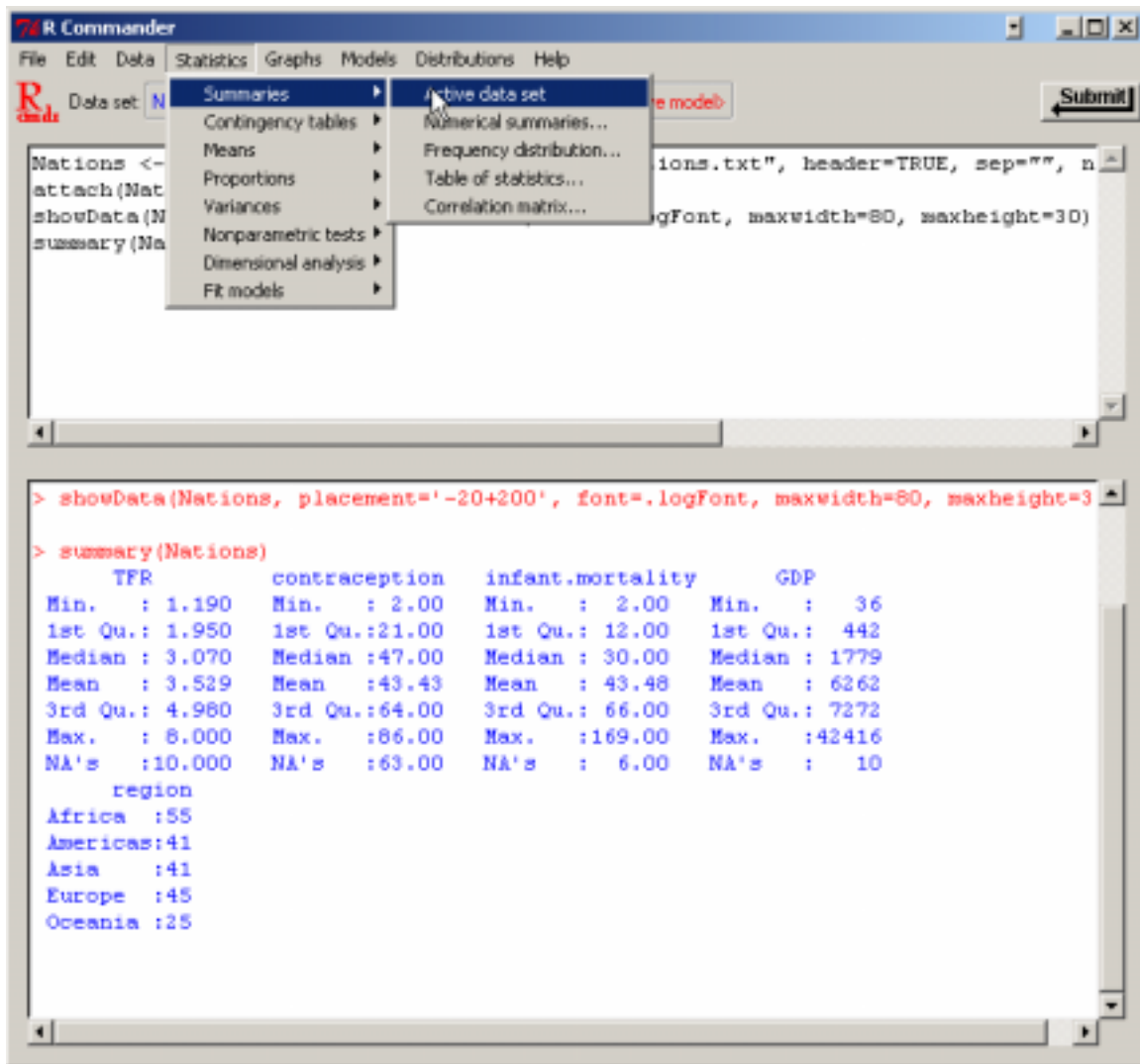


Figure 9: Getting variable summaries for the active data set.

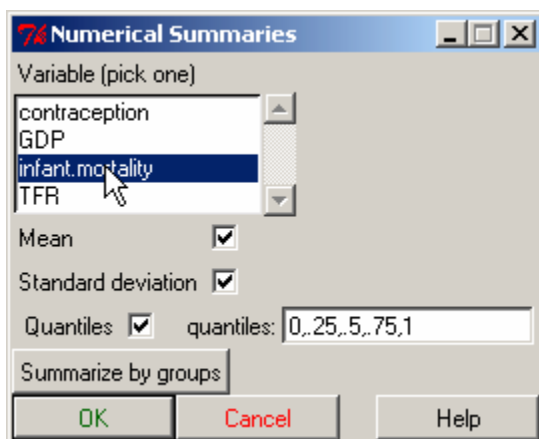


Figure 10: The Numerical Summaries dialog box.

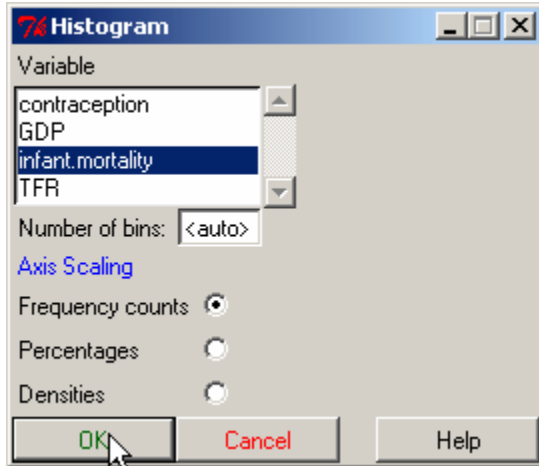


Figure 11: The Histogram dialog.

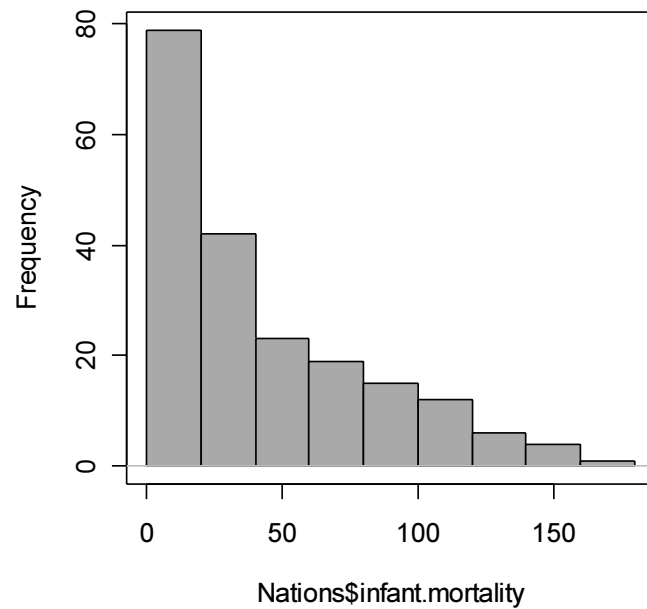


Figure 12: A graphics window containing the histogram for infant mortality.

Saving and Printing Output

You can save text output directly from the *File* menu in the *R Commander*; likewise you can save or print a graph from the *File* menu in an *R Graphics Device* window. It is generally more convenient, however, to collect the text output and graphs that you want to keep in a word-processor document. In this manner, you can intersperse R output with your typed notes and explanations.

Open a word processor such as Word, WordPerfect, or even Windows WordPad. To copy text output from the output window, block the text with the mouse, select *Copy* from the *Edit* menu (or press the key combination *Ctrl-c*, or right-click in the window and select *Copy* from the context menu), and then paste the text into the word-processor window via *Edit* → *Paste* (or *Ctrl-v*), as you would for any Windows application. One point worth mentioning is that you should use a mono-spaced (“typewriter”) font, such as *Courier New*, for text output from R; otherwise the output will not line up neatly.

Likewise to copy a graph, select *File* → *Copy to the clipboard* → *as a Metafile* from the *R Graphics Device* menus; then paste the graph into the word-processor document via *Edit* → *Paste* (or *Ctrl-v*). Alternatively, you can use *Ctrl-w* to copy the graph from the *R Graphics Device*, or right-click on the graph to bring up a context menu, from which you can select *Copy as metafile*.⁹ At the end of your R session, you can save or print the document that you have created, providing an annotated record of your work.

Terminating the R Session

There are several ways to terminate your session. For example, you can select *File* → *Exit* → *From Commander and R* from the *R Commander* menus. You will be asked to confirm, and then asked whether you want to save the contents of the script and output windows. Likewise, you can select *File* → *Exit* from the *R Console* menus; in this case, you will be asked whether you want to save the R workspace (i.e., the data that R keeps in memory); you would normally answer *No*.

Extending the R Commander

As is the case for any R package, a user can modify the code for the `Rcmdr` package and recompile the package. Two features make it possible, however, to modify or add to the `Rcmdr` package without recompiling it:

1. The *R Commander* menus are defined in the plain-text (ASCII) file `Rcmdr-menus.txt`, which resides in the package’s `etc` directory. Modifying this file changes the menus. The format of the file is described below.

⁹ As you will see when you examine these menus, you can save graphs in a variety of formats, and to files as well as to the clipboard. The procedure suggested here is particularly simple, however, and generally results in high-quality graphs.

2. Files with extension (file type) `.R` in the `etc` directory are “sourced” (read into memory) when the *R Commander* starts up. Consequently, functions and variables defined in `.R` files are available in the global environment.

Suppose, for example, that we want to provide a menu-item and dialog box for multivariate Box-Cox transformations to normality. The `car` package, which is one of the packages that `Rcmdr` loads at startup, contains a function to perform the necessary computations, `box.cox.powers` (see Fox, 2002, pp. 111-112). Because none of the existing *R Commander* menus seems appropriate, I will add a *Transform* menu under *Statistics*, with the single item *Multivariate Box-Cox transformations...*. This item will lead to a dialog box to select the variables to be transformed. Finally, I will write a function, named `BoxCox`, to construct the dialog box and invoke `box.cox.powers`.

The modified `Rcmdr-menus.txt` appears in Figure 12 (eliding most of the lines in the file). I believe that the format of the file is largely self-explanatory, but allow me to draw your attention to the following points:

- Each line in the file contains five entries (fields) and defines either a `menu` or a `menu item`.
- Each menu has a “parent” menu; top-level menus, such as *File* or *Statistics*, have `topMenu` as their parent. Menu definition requires two lines: One to create the menu and another to place it under its parent.
- The “operation” field in each line contains the parent menu (for menu creation), `cascade` (for placing a menu under its parent), or `command` (for a menu item that invokes a command).
- The “label” field contains the text that labels a menu or menu item. By convention, menu items leading to dialog boxes have labels ending in ellipses, `. . . .`
- The final field contains the name of a function to be invoked by a menu item, or the name of a menu to be installed.
- The last two fields are empty (`""`) for `menu` (as opposed to `item`) lines.

Note the line in the modified `Rcmdr-menus.txt` file creating `transformMenu` as a child of `statisticsMenu`; the line creating the Box-Cox item under `transformMenu`; and the line cascading `transformMenu` under `statisticsMenu`.

```

# R Commander Menu Definitions

# last modified 30 April 04 by J. Fox

#   type      menu/item      operation/parent  label      command/menu

      menu      fileMenu      topMenu      ""      ""
      item      fileMenu      command      "Load log from file..."      loadLog
      item      fileMenu      command      "Save log..."      saveLog
. . .
      menu      statisticsMenu  topMenu      ""      ""
      menu      summariesMenu  statisticsMenu ""      ""
      item      summariesMenu  command      "Active data set"      summarizeDataSet
. . .
      item      modelsMenu      command      "Linear model..."      linearModel
      item      modelsMenu      command      "Generalized linear model..."      generalizedLinearModel

      menu      transformMenu  statisticsMenu ""      ""
      item      transformMenu  command      "Multivariate Box-Cox transformations..."      BoxCox

      item      topMenu          cascade      "Statistics"      statisticsMenu
      item      statisticsMenu  cascade      "Summaries"      summariesMenu
      item      statisticsMenu  cascade      "Contingency tables"      tablesMenu
      item      statisticsMenu  cascade      "Means"      meansMenu
      item      statisticsMenu  cascade      "Proportions"      proportionsMenu
      item      statisticsMenu  cascade      "Variances"      variancesMenu
      item      statisticsMenu  cascade      "Nonparametric tests"      nonparametricMenu
      item      statisticsMenu  cascade      "Dimensional analysis"      dimensionalMenu
      item      statisticsMenu  cascade      "Fit models"      modelsMenu

      item      statisticsMenu  cascade      "Transform"      transformMenu

      menu      graphsMenu      topMenu      ""      ""
. . .

```

Figure 12: Rcmdr-menus.txt file containing new lines for Statistics → Transform → Multivariate Box-Cox transformations; . . . indicates elided lines

The remaining task is to write the `BoxCox` function. The `Rcmdr` package exports a number of functions and global variables to assist in writing dialogs and performing computations:¹⁰

Variables	
<i>Variable</i>	<i>Contents</i>
<code>.activeDataSet</code>	Name of the active data set; starts as <code>NULL</code> .
<code>.activeModel</code>	Name of the active linear or generalized-linear model; starts as <code>NULL</code> .
<code>.factors</code>	Names of factors in the active data set; length 0 if there are no factors.
<code>.modelNumber</code>	Number of the active statistical model; starts at 0.
<code>.numeric</code>	Names of numeric variables in the active data set; length 0 if there are no numeric variables.
<code>.twoLevelFactors</code>	Names of two-level factors in the active data set; length 0 if there are no two-level factors.
<code>.variables</code>	Names of all variables in the active data set.
Functions	
<i>Function</i>	<i>Purpose</i>
<code>activeDataSet</code>	Returns or sets the name of the active data set.
<code>activeModel</code>	Returns or sets the name of the active model.
<code>doItAndPrint</code>	Executes a command, given as a character string, and echoes the command to the <i>Commander</i> log window.
<code>is.valid.name</code>	Checks that a character string is a valid R name.
<code>listDataSets</code>	Lists names of data frames, by default in the global environment.
<code>listGeneralizedLinearModels</code>	Lists names of <code>glm</code> objects, by default in the global environment.
<code>listLinearModels</code>	Lists names of <code>lm</code> objects, by default in the global environment.
<code>logger</code>	Echoes a character string to the <i>Commander</i> log window without executing it as a command.
<code>justDoIt</code>	Executes a character string without echoing it to the <i>Commander</i> log window.

¹⁰ There are, in addition, several statistical functions exported by the `Rcmdr` package: `colPercents` and `rowPercents` for nicely formatted percentage tables; `reliability` to estimate the reliability of composite scales; `partial.cor` for matrices of partial correlations; `stem.leaf` for high-quality stem-and-leaf displays (generously made available to me by Peter Wolf); `Hist` for enhanced histograms; and `scatter3d` for dynamic 3D scatterplots.

The dialog box to be created is very simple: It should have a variable list from which one or more numeric variables are to be selected, along with *OK*, *Cancel*, and *Help* buttons. A generally simple way to proceed is to find an Rcmdr dialog that is similar and modify it, rather than creating code from scratch. In this case, I started with the code for the `scatterPlotMatrix` dialog, removing unnecessary elements and making small changes. The resulting code is shown in Figure 13 . This example assumes some familiarity with Tcl/Tk and the `tcltk` package. An illustrative dialog box appears in Figure 14.

```
BoxCox <- function(){
  if (activeDataSet() == FALSE) {
    tkfocus(.commander)
    return()
  }
  if (length(.numeric) < 1){
    tkmessageBox(
      message="There no numeric variables in the active data set.",
      icon="error", type="ok")
    tkfocus(.commander)
    return()
  }
  top <- tktoplevel()
  tkwm.title(top, "Box-Cox Transformations")
  variablesFrame <- tkframe(top)
  variablesBox <- tklistbox(variablesFrame, height=min(4, length(.numeric)),
    selectmode="multiple", background="white", exportselection="FALSE")
  variablesScroll <- tkscrollbar(variablesFrame,
    repeatinterval=5, command=function(...) tkyview(variablesBox, ...))
  tkconfigure(variablesBox, yscrollcommand=function(...))
  tkset(variablesScroll, ...)
  for (variable in .numeric) tkinsert(variablesBox, "end", variable)
  onOK <- function(){
    variables <- .numeric[as.numeric(tkcurselection(variablesBox)) + 1]
    if (length(variables) < 1) {
      tkmessageBox(message="You must select one or more variables.",
        icon="error", type="ok")
      tkdestroy(top)
      BoxCox()
      return()
    }
    if (.grab.focus) tkgrab.release(top)
    tkdestroy(top)
    command <- paste("box.cox.powers(na.omit(cbind(",
      paste(variables, collapse=","), "))", sep="")
    doItAndPrint(command)
    tkfocus(.commander)
  }
  onCancel <- function() {
    if (.grab.focus) tkgrab.release(top)
    tkfocus(.commander)
    tkdestroy(top)
  }
  buttonsFrame <- tkframe(top)
  OKbutton <- tkbutton(buttonsFrame, text="OK", fg="darkgreen", width="12",
    command=onOK, default="active")
  cancelButton <- tkbutton(buttonsFrame, text="Cancel", fg="red", width="12",
    command=onCancel)
```

```

onHelp <- function() {
  if (.Platform$OS.type != "windows") if (.grab.focus)
tkgrab.release(top)
  help(box.cox.powers)
}
helpButton <- tkbutton(buttonsFrame, text="Help", width="12",
command=onHelp)
tkgrid(tklabel(top, text="Select variables (one or more)"), sticky="w")
tkgrid(variablesBox, variablesScroll, sticky="nw")
tkgrid(variablesFrame, sticky="nw")
tkgrid(OKbutton, cancelButton, tklabel(buttonsFrame, text="      "),
helpButton, sticky="w")
tkgrid(buttonsFrame, sticky="w")
for (row in 0:2) tkgrid.rowconfigure(top, row, weight=0)
for (col in 0:0) tkgrid.columnconfigure(top, col, weight=0)
.Tcl("update idletasks")
tkwm.resizable(top, 0, 0)
tkgrid.configure(variablesScroll, sticky="ns")
tkgrid.configure(helpButton, sticky="e")
tkbind(top, "<Return>", onOK)
if (.double.click) tkbind(top, "<Double-ButtonPress-1>", onOK)
tkwm.deiconify(top)
if (.grab.focus) tkgrab.set(top)
tkfocus(top)
tkwait.window(top)
}

```

Figure 13: A function to create a dialog box for Box-Cox transformations. The function is placed in the file `BoxCox.R` in the `Rcmdr etc` directory.

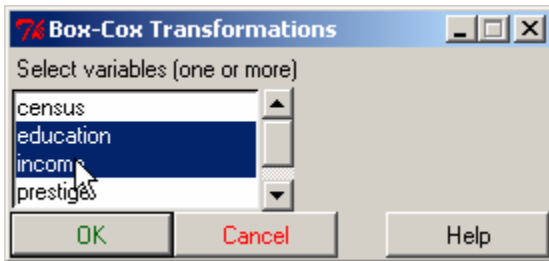


Figure 14: An illustrative dialog box produced by the code in Figure 13.

Notice the use of the `doItAndPrint` to execute the command, send the command to the log, and send the command and output to the output window. This approach will work in most cases.

The code for this example is in the file `BoxCox.demo` in the `etc` directory of the `Rcmdr` package. Rename the file to `BoxCox.R` to activate it. Likewise, the `Rcmdr-menus.txt` file distributed with the package contains commented-out lines for the example; remove the comment characters (`#`) from the beginnings of these lines to activate them.

Some Suggestions for Instructors

The social-science students that I encounter in introductory statistics classes often have difficulty installing and configuring software. I imagine that this situation varies with discipline and locale, but I also expect that it is reasonably common. I assume here that students will be using R and the R Commander under Windows, but it should not be hard to transpose these suggestions to other operating systems.

I distribute to students a CD-ROM with a live, installed version of R, including all necessary packages, and configured to open R in SDI mode, to load the `Rcmdr` package at startup, and to use compiled HTML help in R. Students can simply double click on the file `Run-R.bat` in the root directory of the CD to start R. This batch file contains a single line:

```
start rw1090\bin\Rgui.exe
```

There is another batch file in the root directory of the CD for installing R on the student's computer. This batch file runs the R installer (located in the R directory of the CD) and then copies over the necessary installed packages and configuration files. The contents of `Install-R.bat` are as follows (note that some batch commands are wrapped to fit on the page but are single lines in the file):

```
R\rw1090.exe

xcopy rw1090\library\abind\*.*
      "c:\program files\R\rw1090\library\abind\" /I /E /Y

xcopy rw1090\library\car\*.*
      "c:\program files\R\rw1090\library\car\" /I /E /Y

xcopy rw1090\library\effects\*.*
      "c:\program files\R\rw1090\library\effects\" /I /E /Y

xcopy rw1090\library\lmtest\*.*
      "c:\program files\R\rw1090\library\lmtest\" /I /E /Y

xcopy rw1090\library\multcomp\*.*
      "c:\program files\R\rw1090\library\multcomp\" /I /E /Y

xcopy rw1090\library\mvtnorm\*.*
      "c:\program files\R\rw1090\library\mvtnorm\" /I /E /Y

xcopy rw1090\library\Rcmdr\*.*
      "c:\program files\R\rw1090\library\Rcmdr\" /I /E /Y

xcopy rw1090\library\relimp\*.*
      "c:\program files\R\rw1090\library\relimp\" /I /E /Y

xcopy rw1090\library\rgl\*.*
      "c:\program files\R\rw1090\library\rgl\" /I /E /Y
```

```
xcopy rw1090\etc\Rconsole "c:\program files\R\rw1090\etc\Rconsole" /Y
xcopy rw1090\etc\Rprofile "c:\program files\R\rw1090\etc\Rprofile" /Y
```

This procedure is a bit awkward, and I expect that it will eventually be superseded by facilities for customizing the R Windows installer.

There is a `ReadMe.txt` file in the root directory of the CD with the following contents:

This CD/ROM is intended for Windows 9x, ME, NT, 2000, and XP systems.

The CD/ROM contains the following files and directories:

- o The file `Install-R.bat` will install the R software on your computer and configure it for use in the course. Double-click on the file in the Windows Explorer to initiate the installation process. Take all of the defaults in the R installer. In particular, make sure that R is installed into the directory `C:\Program Files\R\rw1090\`.
- o The directory `rw1090\` contains a pre-installed copy of R that can be run directly from the CD/ROM. Double-click on the file `Run-R.bat` in the Windows Explorer to run R from the CD/ROM.
- o The directory `R\Packages\` contains zip files for all of the packages on CRAN (the Comprehensive R Archive Network).

Finally, the `Rprofile` file has the following contents

```
options(chmhelp=TRUE)
library(Rcmdr)
```

while the `Rconsole` file contains the line

```
MDI = no
```

along with its other, unmodified, contents.

References

- Dalgaard, P. “A primer on the R-Tcl/Tk package.” *R News*, 1(3):27-31, 2001.
- Dalgaard, P. “Changes to the R-Tcl/Tk package.” *R News*, 2(3):25-27, 2002.
- Fox, J. *An R and S-PLUS Companion to Applied Regression*. Newbury Park, CA: 2002.
- Ihaka, R. and R. Gentleman. R. “A language for data analysis and graphics.” *Journal of Computational and Graphical Statistics*, 5:299-314, 1996.
- Moore, D.S. *The Basic Practice of Statistics, Second Edition*. New York: Freeman, 2000.
- Moore, D.S. *The Basic Practice of Statistics, Third Edition*. New York: Freeman, 2004.
- R Development Core Team. *R: A Language and Environment for Statistical Computing*. Vienna: R Foundation for Statistical Computing, 2004.