Data Analysis Through Modeling:
Thinking and Writing in Context
Supplement: Using R and R Commander

Kris Green and Allen Emerson

Fall 2014 Edition¹

¹©2014 Kris H. Green
# Contents

1 Downloading and Installing R  

2 The Basic R Interface  
   2.1 Saving Files in R  
   2.2 Commenting Your R Scripts  
   2.3 Upgrading R  
   2.4 Copying Output from R to Other Programs  

3 Extending R with Packages  
   3.1 Installing and loading packages  
   3.2 The R Commander Package: 
       \texttt{Rcmdr}  
   3.3 Adding \texttt{Rcmdr} to the Auto-start features of R  
   3.4 The \texttt{reshapeGUI} Package  
   3.5 Other useful packages for this text  

4 Creating Data Sets in R Commander  

5 Importing Excel Files in R Commander  

6 Manipulating Data  
   6.1 Converting Numeric Data to Factor Data  
   6.2 Re-Ordering Factor Levels  
   6.3 Constructing a New Variable from Old Variables  
   6.4 Transforming Data to Make Nonlinear Models  

7 Computing Statistics in R Commander  
   7.1 Summary Statistics  
   7.2 Correlation  
   7.3 Z-scores  

8 Making Pivot Tables in R with \texttt{reshapeGUI}  
   8.1 Planning the Table  
   8.2 Melting Variables  
   8.3 Casting the Data Frame  
   8.4 Advanced Features  

iii
9 Graphs and Charts
  9.1 Boxplots ........................................ 27
  9.2 Histograms ........................................ 28
  9.3 Scatterplots ....................................... 28
  9.4 3D Scatterplots .................................... 30

10 Regression Models
  10.1 Simple Linear Models .............................. 33
  10.2 Diagnostic Graphs ................................. 34
  10.3 Multiple Regression Models ....................... 36
  10.4 Reducing Full Regression Models ................. 36
  10.5 Regression Models with Categorical Data ........ 36
  10.6 Regression Models with Interaction Terms ........ 36
  10.7 Stepwise Regression .............................. 37
  10.8 Nonlinear Regression Models ..................... 37

11 Using uniroot ....................................... 39

12 Using lpSolve ....................................... 41

Preliminary Notes

All examples in this guide use the data in ”C10 Enpact Data.xls”.
CHAPTER 1

Downloading and Installing R

All the files and information you need to download R can be found online at:

http://www.r-project.org/

You can also search online for “R statistical package download”. R is available for Windows, Mac, and most other platforms. It is completely free and easy to update and extend to new capabilities. This software allows users to do sophisticated statistical analysis and mathematical modeling, similar to other, commercial packages.
CHAPTER 2

The Basic R Interface

The basic R interface (figure 2.1) is not very exciting. It consists primarily of a menu bar, a box with a bunch of text in it, and blank space. There are also some icons/buttons of the more commonly used commands just below the menus.

Menu options:

- File: Lets you load files and save files and print stuff.
- Edit: Lets you select and copy and so forth.
- View: Lets you choose which features of the R interface are visible.
- Misc: Allows you to interrupt computations and adjust how R is working.
- Packages: Lets you load and install new packages to extend R’s basic capabilities.
- Windows: Controls the various windows.
- Help: Accesses the help features.

But the majority of working with R comes from typing commands into the “R Console”. R is a command line program, so many of its powerful capabilities are not accessed through menus, but by typing the commands into the console. For the most part, we will use the package “R Commander” which wraps all this in a graphical user interface (GUI) so that you can mostly point-and-click to run the commands you want.

2.1 Saving Files in R

Also note that when saving work in R there are basically three things you can save:

1. Data file: You can save the active data file(s) to the computer in their current state.

2. Workspace: You can save the console image - a list of all the commands that you have run since starting R. Then you can reload that workspace image and run the commands again (by selecting them and hitting Control plus the R key) to re-create everything.

3. Output: You can also save the output you have generated - all the charts and statistics, etc. - in a file.
2.2 Commenting Your R Scripts

As you run commands from R Commander’s menu options, you will also see the script commands needed; these are output in the script window of the R Commander interface. You can also directly type commands into the script window. Even more importantly, though, is that you should add comments to your script file to remind you what the scripts are for. Comments all start with a “hashtag” symbol: #. Get in the habit of adding comments at the beginning of the file to explain the problem the file will help you solve, and add additional comments throughout the file to explain what you each command in the script file does.

2.3 Upgrading R

The following script (courtesy of the folks at http://www.r-statistics.com/) will detect the current version of R and update it to the latest version, if necessary, if you are working on a Windows machine.

```r
# installing/loading the latest installr package:
install.packages('installr'); require(installr) # load / install+load installr
```
updateR()

2.4 Copying Output from R to Other Programs

Copying graphs/charts is easy. First right-click on the graph. Select “Copy as metafile”. Then simply paste (CTRL + V) in the Word document. You could also copy an image of the graph as a bitmap or other option, but the metafile option will make the graph more useable.

Copying text output and tables is a little harder. If you simply select the text you want by highlighting it in the Output Window, then copy it (using CTRL + C or the Edit/Copy menu) and paste it into another document, say a Word document, you will not be happy with the results. None of the formatting will come with the copy process. In order to retain the table formatting, you need to complete the following steps, shown as a script in figure 2.2

1. Assign the table (or whatever) to a data frame, by adding “VarName <-” in front of the command that generated the output in the Script Window (see Figure 2.2) and running the command with “CTRL + R”. Note that you will also need to remove any line breaks in the script command or add a plus sign ”+” at the beginning of any additional lines of script.

2. Copy the named data frame to the clipboard using a script and run the command with “CTRL + R” (see the Script Window in Figure 2.2)

3. Paste the clipboard into your Word document

4. Select the text

5. Go to “Insert/Table...” and select the “Convert text to table” option

6. Make sure the options for the table are set to “Fixed column width” and “Separate text at commas”, then hit “OK”
Figure 2.2: Scripts needed to output data to a Word document.
CHAPTER 3

Extending R with Packages

3.1 Installing and loading packages

Installing packages in R is relatively easy. First go to the Packages/Install packages... menu option. You will be presented with a list of locations where these packages are stored. It is usually best to select a location relatively close to you, or at least one on the same continent. After selecting a site, you will then see a list of packages available. Once you choose one (in this case select Rcmdr), the rest will happen automatically.

Note: You might be asked whether you want to use a “personal library”. If so, select “Yes”, and then allow it to create a new library directory by answering “Yes” to the next question.

Once you have installed a package, you can load the package with the library command. For example, to load the “Rcmdr” package, just go into the “R Console” and type

library(Rcmdr)

When you first run a new package, you may get a message explaining that the package requires other packages to run. If you select “Yes”, R will automatically download and install the needed packages the first time (it should only need to do this the first time you run a newly installed package.)

3.2 The R Commander Package: Rcmdr

This package provides you with a menu-driven interface to use most of the common statistical tools. The window is divided into five areas. From the top of the window to the bottom (shown in figure 3.1), these are:

1. The menu options available for performing various analyses.

2. Information about the currently active data set and options for modifying and displaying it.

3. The Script Window: When you use a menu to activate a command, the computer code for the command is printed to the script window.
4. The Output Window shows the results of the commands you run.

5. The Message Window shows any error messages or other system information from your commands.

![Figure 3.1: Screen image of R Commander interface.](image)

For more information, try http://cran.r-project.org/doc/contrib/Karp-Rcommander-intro.pdf

Also, in most of the dialog boxes of R Commander, there is a help button that will pull up a specific web page on the feature.

### 3.3 Adding Rcmdr to the Auto-start features of R

If you are working from home, you may want to bypass typing `library(Rcmdr)` every time you start R. You can edit the `rprofile.site` file in the `etc` directory of R by adding the following lines:

```r
local(
  old <- getOption("defaultPackages")
  options(defaultPackages = c(old, "Rcmdr"))
)
```
3.4 The reshapeGUI Package

The reshapeGUI package offers an interface for cross-sectioning data similar to the way a Pivot Table in Microsoft Excel works. First, install the package by going to the “Packages/Install packages...” as you did with the Rcmdr package. Once it is installed, you can run this package with the `library(reshapeGUI)` command. You will then need to:

1. Invoke the interface
2. Load the data file
3. Melt the data from its raw form
4. Cast the data into the form you want

To invoke the interface, first type (in the R interface)

```r
library(reshapeGUI)
```

The first time you run the `library(reshapeGUI)` R will probably need to install some additional packages (possibly one called GTK+). This may require you to close out of R and restart it once the installation is complete. You can then activate the GUI (shown in figure 3.2) with

```r
reshapeGUI()
```

Using the interface to cross-section your data is described later.

3.5 Other useful packages for this text

You will also want to install the lpSolve package for linear programming. It is similar to Excel’s Solver tool.

In addition, the uniroot package allows you to find roots of equations like Excel’s Goal Seek tool.
Figure 3.2: Screen image of the reshapeGUI interface.
CHAPTER 4

Creating Data Sets in R Commander

To create a new data set using R Commander, use the “Data/New Data Set...” to begin. A dialog box will require you to enter a name for the new data set. When you name the data set, you can only use alphanumeric characters or periods, and the name cannot start with a number. You cannot use spaces, other punctuation, or strange characters. Next, you will see a blank data set organized in rows and columns. If you need more columns, you can resize the window by dragging the edges. You can then click on a column heading (like “var1”) to give each variable a name and select whether it is a numerical or categorical (character) variable (see figure 4.1). The same naming rules for data sets apply to variable names.

Click on a specific cell in the table to enter data. Hitting “Enter” will move your cursor to the next cell down. You can right-click on the columns to adjust their sizing. Manual data entry is somewhat limited. For example, in Excel, you could easily create a long series of numbers like 2, 4, 6, 8, ..., 500 by entering the first few entries in the series, highlighting them and using the fill handle to drag down until you get to the last number in the series. There is no such tool in R or R Commander, so you will either have to enter all the numbers in the series manually, or use another tool (like Excel) to construct the series and then import this file into R.

Your other option for creating data that follows a pattern is through scripts. See section 2.1.5 of http://maths-people.anu.edu.au/johnm/r/usingR.pdf
Figure 4.1: Window for creating a new interface in R Commander
CHAPTER 5

Importing Excel Files in R Commander

The first task, before attempting to load a data set into R through the R Commander interface is to clean the data. R and R Commander expect the data to be formatted a certain way.

1. The top row of the data file should consist of the names of the variables in the data.
2. Each variable name should consist only of letters and numbers; no spaces or other symbols.
3. Each row should be the observations for each variable.
4. If there is data missing, use “NA” for the observation; missing data will cause R to load the data file incorrectly.

Once the data file is prepared, you can load the data using the R Commander interface. Go to “Data/Import Data.../from Excel file...” as shown in figure 5.1. A dialog box will then ask you to name the data file - give it a descriptive name, so that if you are working with several files you can easily distinguish them. Then browse through to find the file you want. Once it is loaded, you can view the file with the “View Data Set” button or edit it with the “Edit Data Set” button.

Also note that this defaults to import files with the “.XLS” extension. If your file is in the newer format, “.XLSX”, you can either open the file in Excel and save it as an older workbook format, or you can simply change the option from ”Excel files (.xls)” to ”Excel 2007 Files (.xlsx)” with the pull down menu in the dialog box where you search for the data file.

This same series of steps will allow you to import data that is formatted for other statistical software, like SPSS.

You may need to recode some of your data, or use the data manipulation tools to make certain that the data is interpreted correctly.
Figure 5.1: Importing Excel files with R Commander.
CHAPTER 6

Manipulating Data

6.1 Converting Numeric Data to Factor Data

Often, a categorical variable (factor) is coded using numbers. For example, you might use 0 to code for female and 1 for male. Or you might use numerical codes for different jobs at a company. In R (and R Commander) any data coded as a number is treated as a numerical variable. If you want to treat these variables as categorical, you will need to convert them. One way to do this is to use “Data/Manage variables in active data set.../Convert numeric variables to factors...” This dialog box is shown in figure 6.1.

You can then select which variable to recode, and give that variable a new name. Clicking on “OK” will give you a new dialog box with a list of the values of the variable you want to recode. For each, enter a new code that is more descriptive, for example “M” for male and “F” for female. You can even use entire words.

6.2 Re-Ordering Factor Levels

By default, R Commander (and R) will assume that the factors of a categorical variable should be organized alphabetically. This is rarely useful. For example, if your factors are “Hi”, “Med”, and “Low”, the default will order them “Hi”, “Low”, “Med” when using this factor to summarize data. Thus, you will sometimes need to tell R what order they should be interpreted in. Click on “Data/Manage data in active data set/Re-order factor levels”. Then select a factor from the list. This will bring up a dialog box with all the factors listed. Next to each is a text box to enter the order in which they should appear.

6.3 Constructing a New Variable from Old Variables

You can easily construct a new variable from existing data. Go to “Data/Manage variables in active data set” and choose “Compute new variable” to get the dialog box shown in figure 6.2. For example, if we want to subtract 18 from the age of each person to focus on the number of years each person has been an adult, we can select “Age” as the base variable, then naming the new variable (like “YrsAdult”) and entering a formula, like “Age-18”. Clicking
6.4 Transforming Data to Make Nonlinear Models

While we have listed this separately, this is actually just a specific case of the previous topic, “Constructing new variables from existing variables.” For example, you could compute the natural logarithm of a variable by entering “log(Age)” as the formula. Or you could use one of the options listed below:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>log(x)</td>
<td>computes the natural log</td>
</tr>
<tr>
<td>exp(x)</td>
<td>exponentiation</td>
</tr>
<tr>
<td>sqrt(x)</td>
<td>square root</td>
</tr>
<tr>
<td>x^2</td>
<td>square the variable</td>
</tr>
</tbody>
</table>
Figure 6.2: Dialog box for computing a new variable from existing variables.
CHAPTER 7
Computing Statistics in R Commander

7.1 Summary Statistics

Once you have an active data set (either importing one from a file or entering your own) you can create a table of summary statistics. Go to “Statistics/Summaries/Numerical Summaries...” to activate the dialog box shown in Figure 7.1. You can then select which variable(s) and which statistics you want. The options include: mean, standard deviation, interquartile range, coefficient of variation, skewness, kurtosis, and quartiles.

You can also summarize the data by groups. For example, you could compute separate statistics for all the male versus female employees, or summarize by job type. This is a simple way to create an analysis similar to a one-way pivot table in Excel.

7.2 Correlation

To compute a correlation matrix in R Commander, go to “Statistics/Summaries/Correlation matrix...” to get the dialog box shown in Figure 7.2. Next select the numerical variables that you wish to correlate and select one of the options. The standard correlation is the Pearson product-moment. But you can also do various non-parametric tests with the other options.

7.3 Z-scores

Computing z-scores for a variable is straightforward. Go to “Data/Manage variables in active data set” and select “Standardize variables”. Choose one of the variables in the list, and hit “OK”. If you now view the active data set, you will see a new column of data. The name at the top of the column will the original variable name with “Z.” in front of it.
Figure 7.1: Dialog box for computing summary statistics with R Commander.
Figure 7.2: Dialog box for computing correlations among numerical variables.
CHAPTER 8

Making Pivot Tables in R with reshapeGUI

This is done in three stages: Planning the table, which is best done on paper to envision what you want to look at and how you want to cross-section the data; Melting the variables, to condense the cross-section variables from the data variables; and Casting the Data Frame, where you construct the table and pour the melted data into it to take shape. Note that at each stage of this, a new data set will be constructed by the tool.

8.1 Planning the Table

Figure out which variables you want to measure, and how you want to cross section the data. For example, if you want to look at average salaries cross-sectioned by gender or other variables in the data, then you are measuring the salary variable, and all other variables are available for cross-sectioning.

It may help to draw the table out on paper, to be sure that you know what you are planning to do. Once you have a good picture and the GUI is activated, you can load a data set into the GUI’s active frame. Click on the name of the data set (e.g. “EnPact”) to view the data in the lower portion of the GUI, which is shown in figure 8.1.

Note that pivot tables of more than two dimensions can be created, and you can include more than one variable in each dimension. But tables like this get very large and complicated, so at first, you probably want to stick with two dimensional tables. It is also worth noting that in Microsoft Excel’s pivot tables, you can put the same variable from your data in multiple places (e.g. as a row variable and a data variable). That is not possible using the reshapeGUI interface.

8.2 Melting Variables

Assuming you have a picture of that table you want, the first task is to “melt” all the variables that you will not be measuring but that you might want to use for cross-sectioning the data. Once you have the data loaded, click on the “melt” button in the interface to bring up the tools for melting the data, shown in figure 8.2. Initially, the melt interface will look as shown below, with all of the variables in the data listed in the center section of the upper half of the interface which is labeled “Unusued Variables”.

23
Next, you need to move variables to the “ID Variables” and “Measure Variables” lists. You do this by clicking on the variable you want to move, then using the arrow buttons “< −” and “− >” to move the variables to the other lists. Move all the potential cross-section variables to the “ID Variables” list and all the variables you want to measure to the “Measure Variables” list. Any identification variables or other variables not being used for cross-sectioning or measuring can be left in the “Unused Variables” list.

In the bottom half of the melt interface you will see a preview of the melted data. You can click “Preview” to refresh the sample. When you are done, click “Execute” to create the melted data.

### 8.3 Casting the Data Frame

To cast the pivot table, start on the main tab of the reshapeGUI interface. If you are still on the “melt” tab, simply click on the “Data Selection” tab. Then choose a data frame of melted data by clicking on it and click “melt”. You will see the interface shown in figure 8.3. You can then use the pull-down menus to select variables from the list of melted variables for the different dimensions of the table. For simple tables, you may only need a single dimension
8.4. ADVANCED FEATURES

Figure 8.2: The Melt tab of reshapeGUI.

(e.g. measuring average salary for each job grade) or two dimensions (e.g. average salary versus job grade and education). You can also use the pull-down menu under “aggregating function” the select the way the measured variable is displayed, for example as an average or standard deviation.

When you have finished setting up the table, click “execute” to generate the table you have designed and laid out.

8.4 Advanced Features

Once you have cast the data into a table, you can select it and manipulate it using the “ddply” function. This allows you to split the table into pieces, work with each piece using different mathematical operations, and then re-combine the pieces into a single table.

You can also recast the same data multiple times to make different tables.
Figure 8.3: The Cast tab of reshapeGUI.
CHAPTER 9

Graphs and Charts

9.1 Boxplots

Standard boxplots are created by selecting “Graphs/Boxplot...” and choosing a numerical variable from the active data set, shown in figure 9.1.

![Boxplot dialog box](image)

Figure 9.1: Dialog box for generating a boxplot.

You can also create side-by-side boxplots, breaking down one variable into several box-
plots, based on a second, categorical variable (factor). To do this, click the “Plot by groups...” option in the dialog box and select a factor variable.

9.2 Histograms

To plot a histogram, select the “Graphs/Histogram...” option from the menu. Then select a numerical variable from the active data set in the dialog box shown in figure 9.2. Clicking “OK” will produce a histogram with an automatically determined number of bins. You can control the number of bins yourself by entering a number in the box marked “Number of bins”.

![Figure 9.2: Dialog box for generating a histogram.](image)

9.3 Scatterplots

To generate a single scatterplot, use the “Graphs/Scatterplot...” menu. The dialog box that appears (see figure 9.3) has a number of options to control the content and appearance of the plot. Most importantly, you first need to select the independent (X) and dependent (Y) variables.
9.3. SCATTERPLOTS

You then have the following options:

- **Identify points**: labels each point with an identifier
- **Jitter x/y variable**:
- **Log x/y axis**: Used to create logarithmic scales for one or both axes
- **Marginal boxplots**: Adds boxplots of the variables to the graph, outside the margins
- **Least-squares line**: Adds a solid least-squares regression line to the plot
- **Smooth line**: Adds a smoothed “average” line to the plot. **You should de-select this for most plots.**
- **Show spread**: Shows a confidence interval around the smoothed line. **You should de-select this for most plots.**
- **Span for smooth**: Controls how smooth the line is.
- **Point size, Axis text size, Axis-labels text size**: Control the appearance of the text and points in the graph.
• **x-axis/y-axis label:** Allows you to enter text to label the axes

• **Subset expression:**

• **Plot by groups:** Plots different groups of data (e.g. male and female) differently on the same axes as determined by a third variable.

### 9.4 3D Scatterplots

Creating a 3D scatterplot works much like creating a normal scatterplot except you must select two independent variables in the dialog box. First go to “Graphs/3D graphs/3d scatterplot..” to see the main dialog box, as in figure 9.4.

![Figure 9.4: Dialog box for generating a 3D scatterplot](image)

In the box, select the response variable. To select two explanatory variables, hold down the control key and click on each of them separately with the mouse. You will also have the following options:

• **Identify observations with mouse:** Not necessary, most of the time, but lets you hover over a point and get its information.
9.4. 3D SCATTERPLOTS

- Show axis scales: Shows the scale of each axis.
- Show surface grid lines: Shows the “wireframe” of the best-fit surface (if included)
- Show squared residuals: Shows the residuals for the best-fit surface (if included)
- Linear least-squares: Shows the best-fit linear surface to the data as a plane
- Quadratic least-squares: Shows the best-fit quadratic (parabolic, elliptical, hyperbolic) surface to the data
- Smooth regression, Additive regression, df: Options for the regressions in the least-squares fit
- Plot 50% concentration ellipsoid: Shows where the majority of the data is clustered
- Background color: change the background for viewing
- Plot by groups...: Allows you to plot the data differently as a function of a factor variable (such as gender)

Once the graph is constructed, you can click and drag the axes around to view the plot from different angles.
CHAPTER 10

Regression Models

10.1 Simple Linear Models

Constructing a simple linear regression model in R Commander is done through the “Statistics/ Fit Models/ Linear regression...” menu option. This will produce the dialog box shown in figure 10.1.

![Linear Regression Dialog Box](image)

Figure 10.1: Linear regression dialog box.
For simple regression, just select one variable to be the response and another variable to be the explanatory variable. Also, give your model a name; the default name is something like “RegModel.1”. When you are done, hit “OK” and R Commander will produce the model and display the summary measures as shown in figure 10.2.

![Linear regression output](image)

Figure 10.2: Linear regression output.

Note that you can construct more complex models through the “Statistics/ Fit Models/ Linear model...” option. This dialog box (figure 10.3) allows you to create nonlinear models, as well as models with interaction terms (discussed below). If you ever need to produce this summary again, you can use the “Models/Summarize model” menu to select a model and generate the table of information.

### 10.2 Diagnostic Graphs

In order to generate the diagnostic graphs to check the quality of a model, first select one of the models that you have generated. To do this, click on the “Model” option just below the menu options in R Commander and select one model to be active. Next, use the “Models/Graphs” menu to select the diagnostic graphs you want.
10.2. DIAGNOSTIC GRAPHS

Figure 10.3: Linear model dialog box.

- Basic Diagnostic Graphs: Produces four plots for determining the quality of your model. These include a scatterplot of the residuals vs. the fitted values, the normal Q-Q plot, a scale-locatoin plot, and the residuals plotted against their leverage.

- Residual Quartile Comparison Plot: Plots the true model residuals (after standardizing them) against a theoretical distribution.

- Component + Residual Plots: Plots the response variable against each of the explanatory variables.

- Added Variable: Plots the response versus the explanatory variable, includes the regression line, and if you select “Yes” to “Identify points with mouse” you can click on a specific point to display its identifier on the graph. This is useful for identifying outliers in your data.

- Influence Plot: This plots the studentized residuals against the “hat-values” (the fitted values) of the response variable, measured . Each data point is plotted as a circle, where the area of the circle is proportional to Cook’s distance.

- Effect Plot: Not all models have an effect that can be plotted.
10.3 Multiple Regression Models

Constructing a multiple regression model is done through the same menu options as those for constructing simple regression models. The only difference is that when you are selecting the explanatory variables, you can select as many as you want, simply by holding down the control key while you click on each variable name (or click and drag through a block of variables).

10.4 Reducing Full Regression Models

There are two essentially different ways to reduce a full regression model on your own. The first method is to simply re-run the multiple regression routine, selecting one fewer variable than you did the last time, repeating until you have eliminated all the non-significant variables. However, this can be a little tedious, especially if you have a long list of variables to check.

The second method requires you to write short scripts yourself, rather than use R Commander’s menus. Suppose you have produced a regression model with the response variable “Salary” and the explanatory variables Age, EducationLevel, Gender, YrsExp, YrsPrior and named the model “LinearModel.2”. Looking at the summary of the model, we see that Age is not a significant variable \( p = 0.4500 > 0.05 \) so we would like to run the same model, but leave out Age. In the script window, we can run the commands below in the script window to remove Age from the model and summarize the new model.

\[
\text{LinearModel.3} = \text{update(LinearModel.2, } .\sim .-\text{Age)}
\]
\[
\text{summary(LinearModel.3)}
\]

You can also use the \texttt{update} function to add variables to a model by replacing the minus sign with a plus sign. The general format is \texttt{update(model name, what to add or subtract from the model)}. The changes to the model always follow the squiggle (“.~.”).

10.5 Regression Models with Categorical Data

In some statistical packages, modeling with categorical data requires that you construct dummy variables first, then use all but one of those dummy variables (the reference category) in building the model. In R Commander (and R) selecting a categorical variable as one of the explanatory variables is all that you need to do; R will handle the variable correctly behind the scenes.

10.6 Regression Models with Interaction Terms

To include an interaction term in a linear model, you simply multiply the two variable names in your formula. For example, to include an interaction term for Gender and Years of Experience, along with the base variables themselves, build your model by multiplying
those two variables (see figure 10.4 for constructing the model and Figure 10.5 for what the model looks like.) Notice that the model includes Gender (with Female as the base variable), YrsExp, and the interaction term, Gender:YrsExp. To remove one of the base variables, use the update command to build a new model from an existing one (see Section 10.4).

Figure 10.4: Linear model with an interaction term.

10.7 Stepwise Regression

10.8 Nonlinear Regression Models
Figure 10.5: Output of a linear model with an interaction term.
CHAPTER 11

Using uniroot
CHAPTER 11. USING UNIROOT
CHAPTER 12

Using lpSolve