LEARNING MADE EASY

R Projects

for dummies®

A Wiley Brand

Learn a wide range of R applications

Work through R projects and sharpen your skillset

Understand how to execute your own R projects

Joseph Schmuller, PhD

R Projects

by Joseph Schmuller, PhD

for dummies®
A Wiley Brand

R Projects For Dummies®

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Published simultaneously in Canada

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002. For technical support, please visit https://hub.wiley.com/community/support/dummies.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at http://booksupport.wiley.com. For more information about Wiley products, visit www.wiley.com.

Library of Congress Control Number: 2017964027

ISBN: 978-1-119-44618-7; 978-1-119-44617-0 (ebk); 978-1-119-44616-3 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Contents at a Glance

Table of Contents

Introduction

If you're like me, you think the best way to learn is by doing. Don't just read about something — practice it! If you want to be a builder, then build. If you want to be a writer, then write. If you want to be a carpenter, then carpenter. (Yes, that noun and verb are the same. *Carpent* is not a word.)

I based this book on that learning-by-doing philosophy. My objective is for you to expand your R skill set by using R to complete projects in a variety of areas, and to learn something about those areas, too.

Even with those noble intentions, a book like this one can fall into a trap. It can quickly become a cookbook: Use this package, use these functions, create a graphic — and presto, you've finished a project and it's time to move on.

I didn't want to write that book. Instead, beginning in Part 2 (which is where the projects start), each chapter does more than just walk you through a project. First, I show you some background material about the subject area, and then (in most chapters) you work through a scaled-down project in that area to get your feet wet, and then you complete a larger project.

But a chapter doesn't end there. At the end of each chapter, you'll find a Suggested Project that challenges you to apply your newly minted skills. For each of those, I supply just enough information to get you started. (Wherever necessary, I include tips about potential pitfalls.)

Along the way, you'll also encounter Quick Suggested Projects. These are based on tweaks to projects you've already completed, and they present additional challenges to your growing skill set.

One more thing: Every subject area could be the basis for an entire book, so I can only scratch the surface of each one. Chapter 17 directs you toward resources that provide more information.

About This Book

I've organized this book into six parts.

Part 1: The Tools of the Trade

Part 1 is all about R and RStudio. I discuss R functions, structures, and packages, and I show you how to create a variety of graphics.

Part 2: Interacting with a User

The projects begin in Part 2, where you learn to create applications that respond to users. I discuss the `shiny` package for working with web browsers, and the `shinydashboard` package for creating dashboards.

Part 3: Machine Learning

This is the longest part of the book. I begin by telling you about the University of California–Irvine Machine Learning Repository, which provides the data sets for the projects. I also discuss the `rattle` package for creating machine learning applications. The projects cover decision trees, random forests, support vector machines, k-means clustering, and neural networks.

Part 4: Large(ish) Data Sets

The two projects in Part 4 deal with larger data sets than you encounter earlier in the book. The first project is a customer segmentation analysis of over 300,000 customers of an online retailer. A follow-up analysis applies machine learning. The second project analyzes a data set of more than 500,000 airline flights.

Part 5: Maps and Images

Two projects are in Part 5. The first is to plot the location (along with other information) of airports in one of the US states. The second shows you how to combine an animated image with a stationary one.

Part 6: The Part of Tens

The first chapter in Part 6 provides information about useful packages that can help you with future projects. The second tells you where to learn more about the subject areas of this book.

What You Can Safely Skip

Any reference book throws a lot of information at you, and this one is no exception. I intended it all to be useful, but I didn't aim it all at the same level. So if you're not deeply into the subject matter, you can avoid paragraphs marked with the Technical Stuff icon, and you can also skip the sidebars.

Foolish Assumptions

I'm assuming that you

- Know how to work with Windows or the Mac. I don't spell out the details of pointing, clicking, selecting, and other actions.
- Can install R and RStudio (I show you how in Chapter 1), and follow along with the examples. I use the Windows version of RStudio, but you should have no problem if you're working on a Mac.

Icons Used in This Book

You'll find icons in all *For Dummies* books, and this one is no exception. Each one is a little picture in the margin that lets you know something special about the paragraph it sits next to.

TIP

This icon points out a hint or a shortcut that helps you in your work and makes you an all-around better person.

REMEMBER

This one points out timeless wisdom to take with you as you continue on the path to enlightenment.

WARNING

Pay attention to this icon. It's a reminder to avoid something that might gum up the works for you.

TECHNICAL STUFF

As I mention in the earlier section "What You Can Safely Skip," this icon indicates material you can blow past if it's just too technical. (I've kept this information to a minimum.)

Beyond the Book

In addition to what you're reading right now, this product comes with a free access-anywhere Cheat Sheet that presents a selected list of R functions and describes what they do. To get this Cheat Sheet, visit `www.dummies.com` and type **R Projects For Dummies Cheat Sheet** in the Search box.

Where to Go from Here

You can start the book anywhere, but here are a couple of hints. Want to introduce yourself to R and packages? You'll find the info in Chapters 1 and 2. Want to start with graphics? Hit Chapter 3. For anything else, find it in the table of contents or in the index and go for it.

If you're a cover-to-cover reader, turn the page. . . .

1

The Tools of the Trade

IN THIS PART . . .

Learn about R and RStudio

Understand R Functions and Structures

Create your own R Functions

Examine data

Use base R graphics

Graduate to `ggplot2` graphics

IN THIS CHAPTER

- **Getting R and RStudio on your computer**
- **Plunging into a session with R**
- **Working with R functions**
- **Working with R structures**

Chapter 1

R: What It Does and How It Does It

So you're ready to journey into the wonderful world of R! Designed by and for statisticians and data scientists, R has a short but illustrious history.

In the 1990s, Ross Ihaka and Robert Gentleman developed R at the University of Auckland, New Zealand. The Foundation for Statistical Computing supports R, which is growing more popular by the day.

Getting R

If you don't already have R on your computer, the first thing to do is to download R and install it.

You'll find the appropriate software on the website of the Comprehensive R Archive Network (CRAN). In your browser, type this web address if you work in Windows:

```
cran.r-project.org/bin/windows/base
```

Type this one if you work on the Mac:

```
cran.r-project.org/bin/macosx
```

Click the link to download R. This puts a `win.exe` file in your Windows computer or a `pkg` file in your Mac. In either case, follow the usual installation procedures. When installation is complete, Windows users see two R icons on their desktop, one for 32-bit processors and one for 64-bit processors (pick the one that's right for you). Mac users see an R icon in their `Application` folder.

TIP

Both addresses provide helpful links to FAQs. The windows-related one also has the link Installation and Other Instructions.

Getting RStudio

Working with R is a lot easier if you do it through an application called RStudio. Computer honchos refer to RStudio as an IDE (*I*ntegrated *D*evelopment *E*nvironment). Think of it as a tool that helps you write, edit, run, and keep track of your R code, and as an environment that connects you to a world of helpful hints about R.

Here's the web address for this terrific tool:

```
www.rstudio.com/products/rstudio/download
```

Click the link for the installer for your computer's operating system — Windows, Mac, or a flavor of Linux — and again follow the usual installation procedures.

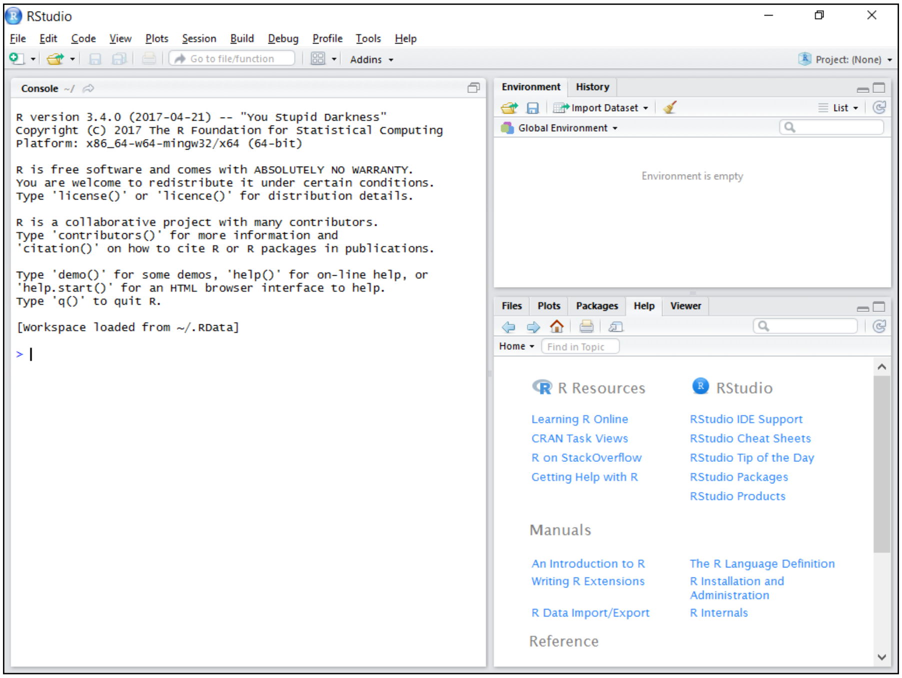
TIP

In this book, I work with R version 3.4.0 and RStudio version 1.0.143. By the time you read this, later versions of both might be available.

After you finish installing R and RStudio, click on your brand-new RStudio icon to open the window shown in Figure 1-1.

The large Console pane on the left runs R code. One way to run R code is to type it directly into the Console pane. I show you another in a moment.

The other two panes provide helpful information as you work with R. The Environment/History pane is in the upper right. The Environment tab keeps track of the things you create (which R calls objects) as you work with R. The History tab tracks R code that you enter.



```
R version 3.4.0 (2017-04-21) -- "You Stupid Darkness"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> |
```

FIGURE 1-1: RStudio, immediately after you install it and click on its icon.

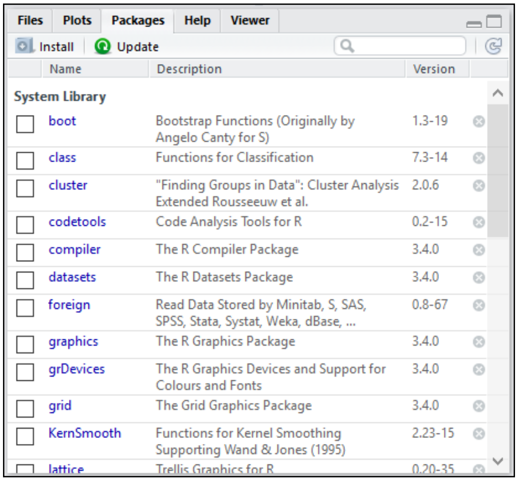
TIP

Get used to the word *object.* Everything in R is an object.The Files/Plots/Packages/Help pane is in the lower right. The Files tab shows files you create. The Plots tab holds graphs you create from your data. The Packages tab shows add-ons (called *packages*) that have downloaded with R. Bear in mind that *downloaded* doesn't mean "ready to use." To use a package's capabilities, one more step is necessary, and trust me — you'll want to use packages.

Figure 1-2 shows the Packages tab. I discuss packages later in this chapter.

The Help tab, shown in Figure 1-3, links you to a wealth of information about R and RStudio.

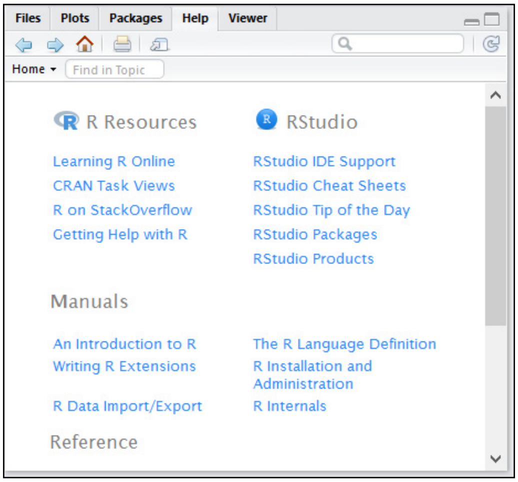
To tap into the full power of RStudio as an IDE, click the icon in the upper right corner of the Console pane. That changes the appearance of RStudio so that it looks like Figure 1-4.



System Library

	Name	Description	Version
□	boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-19
□	class	Functions for Classification	7.3-14
□	cluster	"Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al.	2.0.6
□	codetools	Code Analysis Tools for R	0.2-15
□	compiler	The R Compiler Package	3.4.0
□	datasets	The R Datasets Package	3.4.0
□	foreign	Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, Weka, dBase, ...	0.8-67
□	graphics	The R Graphics Package	3.4.0
□	grDevices	The R Graphics Devices and Support for Colours and Fonts	3.4.0
□	grid	The Grid Graphics Package	3.4.0
□	KernSmooth	Functions for Kernel Smoothing Supporting Wand & Jones (1995)	2.23-15
□	lattice	Trellis Graphics for R	0.20-35

FIGURE 1-2: The RStudio Packages tab.



R Resources

Learning R Online
CRAN Task Views
R on StackOverflow
Getting Help with R

RStudio

RStudio IDE Support
RStudio Cheat Sheets
RStudio Tip of the Day
RStudio Packages
RStudio Products

Manuals

An Introduction to R
Writing R Extensions
R Data Import/Export
The R Language Definition
R Installation and Administration
R Internals

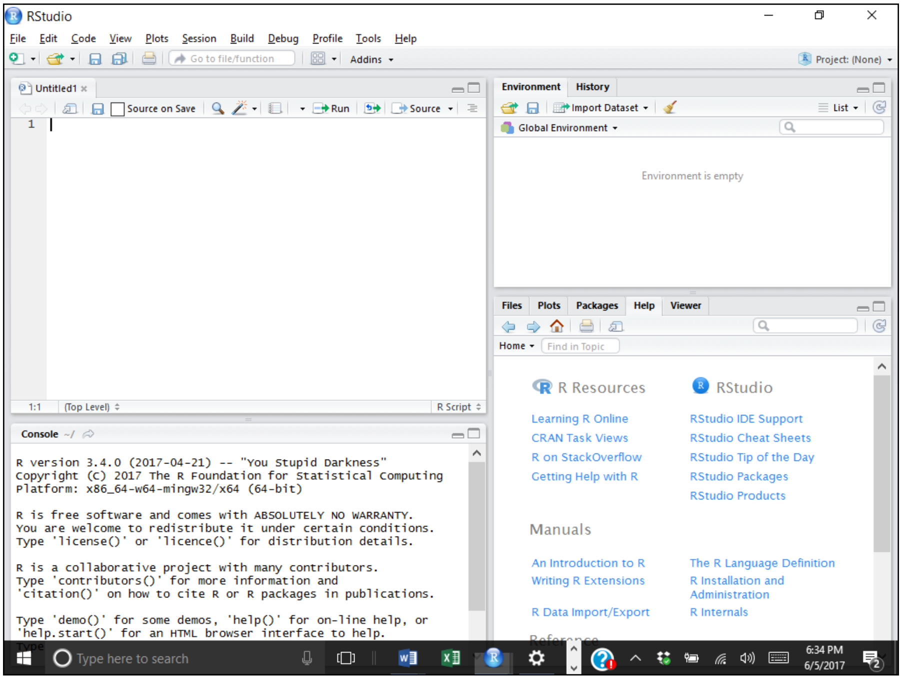
Reference

FIGURE 1-3: The RStudio Help tab.

The Console pane relocates to the lower left. The new pane in the upper left is the Scripts pane. You type and edit code in the Scripts pane by pressing Ctrl+R (Command+Enter on the Mac), and then the code executes in the Console pane.

TIP

Ctrl+Enter works just like Ctrl+R. You can also highlight lines of code in the Scripts pane and select Code ➪ Run Selected Line(s) from RStudio’s main menu.



```
R version 3.4.0 (2017-04-21) -- "You Stupid Darkness"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
```

FIGURE 1-4: RStudio after you click the icon in the upper right corner of the Console pane.

A Session with R

Before you start working, select File➪Save As from the main menu and then save your work file as My First R Session. This relabels the tab in the Scripts pane with the name of the file and adds the `.R` extension. This also causes the filename (along with the `.R` extension) to appear on the Files tab.

The working directory

What exactly does R save, and where does R save it? What R saves is called the *workspace*, which is the environment you're working in. R saves the workspace in the *working directory*. In Windows, the default working directory is

```
C:\Users\<User Name>\Documents
```

On a Mac, it's

```
/Users/<User Name>
```

If you ever forget the path to your working directory, type

```
> getwd()
```

in the Console pane, and R returns the path onscreen.

TIP

In the Console pane, you don't type the right-pointing arrowhead at the beginning of the line. That's a prompt.

My working directory looks like this:

```
> getwd()
[1] "C:/Users/Joseph Schmuller/Documents
```

Note the direction the slashes are slanted. They're opposite to what you typically see in Windows file paths. This is because R uses \ as an *escape character* — whatever follows the \ means something different from what it usually means. For example, \t in R means *Tab key.*

TIP

You can also write a Windows file path in R as

```
C:\\Users\\<User Name>\\Documents
```

If you like, you can change the working directory:

```
> setwd(<file path>)
```

Another way to change the working directory is to select Session➪Set Working Directory➪Choose Directory from the main menu.

Getting started

Let's get down to business and start writing R code. In the Scripts pane, type

```
x <- c(5,10,15,20,25,30,35,40)
```

and then press Ctrl+R.

That puts this line into the Console pane:

```
> x <- c(5,10,15,20,25,30,35,40)
```

As I say in an earlier Tip paragraph, the right-pointing arrowhead (the greater-than sign) is a prompt that R puts in the Console pane. You don't see it in the Scripts pane.

Here's what R just did: The arrow-sign says that x gets assigned whatever is to the right of the arrow-sign. Think of the arrow-sign as R's *assignment operator.* So the set of numbers 5, 10, 15, 20 . . . 40 is now assigned to x.

REMEMBER

In R-speak, a set of numbers like this is a *vector.* I tell you more about this concept in the later section "R Structures."

You can read that line of code as "x gets the vector 5, 10, 15, 20."

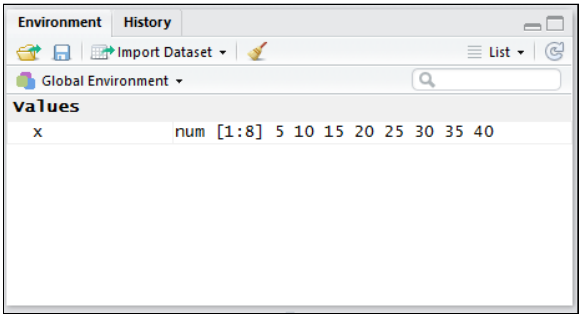
Type **x** into the Scripts pane and press Ctrl+R, and here's what you see in the Console pane:

```
> x
[1]  5 10 15 20 25 30 35 40
```

The 1 in square brackets is the label for the first line of output. So this signifies that 5 is the first value.

Here you have only one line, of course. What happens when R outputs many values over many lines? Each line gets a bracketed numeric label, and the number corresponds to the first value in the line. For example, if the output consists of 23 values and the eighteenth value is the first one on the second line, the second line begins with [18].

Creating the vector x causes the Environment tab to look like Figure 1-5.



```
Values
  x                num [1:8] 5 10 15 20 25 30 35 40
```

FIGURE 1-5: The RStudio Environment tab after creating the vector x.

TIP

Another way to see the objects in the environment is to type **ls()** into the Scripts pane and then press Ctrl+R. Or you can type **> ls()** directly into the Console pane and press Enter. Either way, the result in the Console pane is

```
[1] "x"
```

Now you can work with x. First, add all numbers in the vector. Typing **sum(x)** in the Scripts pane (be sure to follow with Ctrl+R) executes the following line in the Console pane:

```
> sum(x)
[1] 180
```

How about the average of the numbers in vector x?

That would involve typing **mean(x)** in the Scripts pane, which (when followed by Ctrl+R) executes

```
> mean(x)
[1] 22.5
```

in the Console pane.

TIP

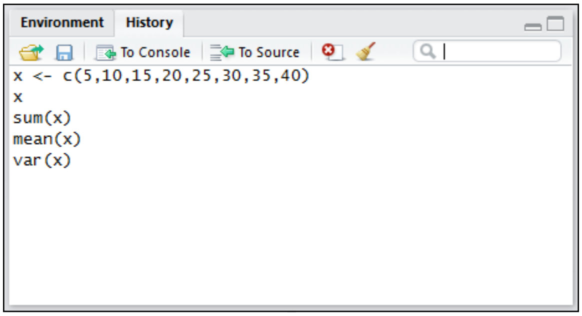
As you type in the Scripts pane or in the Console pane, you see that helpful information pops up. As you become experienced with RStudio, you learn how to use that information.

Variance is a measure of how much a set of numbers differ from their mean. Here's how to use R to calculate variance:

```
> var(x)
[1] 150
```

What, exactly, is variance and what does it mean? (Shameless plug alert.) For the answers to these and numerous other questions about statistics and analysis, read one of the most classic works in the English language: *Statistical Analysis with R For Dummies* (written by yours truly and published by Wiley).

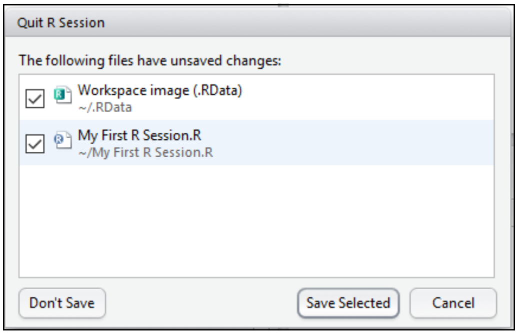
After R executes all these commands, the History tab looks like Figure 1-6.



```
x <- c(5,10,15,20,25,30,35,40)
x
sum(x)
mean(x)
var(x)
```

FIGURE 1-6: The History tab, after creating and working with a vector.

To end a session, select File➪Quit Session from the main menu or press Ctrl+Q. As Figure 1-7 shows, a dialog box opens and asks what you want to save from the session. Saving the selections enables you to reopen the session where you left off the next time you open RStudio (although the Console pane doesn't save your work).



Quit R Session

The following files have unsaved changes:

☑ Workspace image (.RData)
~/.RData

☑ My First R Session.R
~/My First R Session.R

FIGURE 1-7: The Quit R Session dialog box.

REMEMBER

Moving forward, most of the time I don't say "Type this code into the Scripts pane and press Ctrl+Enter" whenever I take you through an example. I just show you the code and its output, as in the `var()` example.

REMEMBER

Also, sometimes I show code with the > prompt, and sometimes without. Generally, I show the prompt when I want you to see R code and its results. I don't show the prompt when I just want you to see R code that I create in the Scripts pane.

R Functions

The examples in the preceding section use `c()`, `sum()`, and `var()`. These are three *functions* built into R. Each one consists of a function name immediately followed by parentheses. Inside the parentheses are *arguments*. In the context of a function, *argument* doesn't mean "debate" or "disagreement" or anything like that. It's the math name for whatever a function operates on.

REMEMBER

Sometimes a function takes no arguments (as is the case with `ls()`). You still include the parentheses.

The functions in the examples I showed you are pretty simple: Supply an argument, and each one gives you a result. Some R functions, however, take more than one argument.

R has a couple of ways for you to deal with multi-argument functions. One way is to list the arguments in the order that they appear in the function's definition. R calls this *positional mapping*.

Here's an example. Remember when I created the vector x?

```
x <- c(5,10,15,20,25,30,35,40)
```

Another way to create a vector of those numbers is with the function `seq()`:

```
> y <- seq(5,40,5)
> y
[1]  5 10 15 20 25 30 35 40
```

Think of `seq()` as creating a "sequence." The first argument to `seq()` is the number to start the sequence *from* (5). The second argument is the number that ends the sequence — the number the sequence goes *to* (40). The third argument is the increment of the sequence — the amount the sequence increases *by* (5).

If you *name* the arguments, it doesn't matter how you order them:

```
> z <- seq(to=40,by=5,from=5)
> z
[1]  5 10 15 20 25 30 35 40
```

So when you use a function, you can place its arguments out of order, if you name them. R calls this *keyword matching*. This comes in handy when you use an R function that has many arguments. If you can't remember their order, use their names, and the function works.

TIP

For help on a particular function — `seq()`, for example — type **?seq**. When you run that code, helpful information appears on the Help tab and useful information pops up in a little window right next to where you're typing.

User-Defined Functions

R enables you to create your own functions, and here are the fundamentals on how to do it.