Making Everything Easier!™

2nd Edition

R

FOR

DUMMIES®

A Wiley Brand

Learn to:

- Use R for data analysis and processing
- Write functions and scripts for repeatable analysis
- Create high-quality charts and graphics
- Perform statistical analysis and build models

Andrie de Vries
Joris Meys

Introduction

Welcome to *R For Dummies,* the book that helps you learn the statistical programming language R quickly and easily.

We can't guarantee that you'll be a guru if you read this book, but you should be able to

- Perform data analysis by using a variety of powerful tools.
- Use the power of R to do statistical analysis and data-processing tasks.
- Appreciate the beauty of using vector-based operations (rather than loops) to do speedy calculations.
- Appreciate the meaning of the following line of code:

```
knowledge <- apply(theory, 1, sum)
```

- Know how to find, download, and use code that has been contributed to R by its very active community of developers.
- Know where to find extra help and resources to take your R coding skills to the next level.
- Create beautiful graphs and visualizations of your data.

About This Book

R For Dummies is an introduction to the statistical programming language known as R. We start by introducing the interface and work our way from the very basic concepts of the language through more sophisticated data manipulation and analysis.

We illustrate every step with easy-to-follow examples. This book contains numerous code snippets, several write-it-yourself functions you can use later on, and complete analysis scripts. All these are for you to try out yourself.

We don't attempt to give a technical description of how R is programmed internally, but we do focus as much on the why as on the how. R has many features that may seem surprising at first, so we believe it's important to explain both how you should talk to R, and how the R engine interprets what you say. After reading this book, you should be able to manipulate your data in the form you want and understand how to use functions we *didn't* cover in the book (as well as the ones we do cover).

This book is a reference. You don't have to read it from beginning to end. Instead, you can use the table of contents and index to find the information you need. We cross-reference other chapters where you can find more information.

Changes in the Second Edition

Since the publication of the first edition, R has kept evolving and improving. To keep the book accurate, we updated the code to reflect any changes in the latest version of R (version 3.2.0). With the feedback from readers, students, and colleagues we could rework some sections to clarify issues and correct inaccuracies. For example, we modified the code to use double quotes instead of single quotes when using text strings. We also refer to the fundamental units of lists as components, rather than elements.

The new `rfordummies` package contains code examples in the book. Read all about it in Appendix B.

R and RStudio

R For Dummies can be used with any operating system that R runs on. Whether you use Mac, Linux, or Windows, this book will get you on your way with R.

R is more a programming language than an application. When you download R, you automatically download a console application that's suitable for your operating system. However, this application has only basic functionality, and it differs to some extent from one operating system to the next.

RStudio is a cross-platform application, also known as an Integrated Development Environment (IDE) with some very neat features to support R. In this book, we don't assume you use any specific console application. However, RStudio provides a common user interface across the major operating systems. For this reason, we use RStudio to demonstrate some of the concepts rather than any specific operating-system version of R.

Conventions Used in This Book

Code snippets appear like this example, where we simulate 1 million throws of two six-sided dice:

```
> set.seed(42)
> throws <- 1e6
> dice <- replicate(2,
+                   sample(1:6, throws, replace = TRUE)
+ )
> table(rowSums(dice))

     2     3     4      5      6      7      8
 28007 55443 83382 110359 138801 167130 138808
     9    10    11    12
110920 83389 55816 27945
```

Each line of R code in this example is preceded by one of two symbols:

- **>:** The prompt symbol, >, is not part of your code, and you should not type this when you try the code yourself.
- **+:** The continuation symbol, +, indicates that this line of code still belongs to the previous line of code. In fact, you don't have to break a line of code into two, but we do this frequently, because it improves the readability of code and helps it fit into the pages of a book.

Lines that start without either the prompt or the continuation symbol are output produced by R. In this case, you get the total number of throws where the dice added up to the numbers 2 through 12. For example, out of 1 million throws of the dice, on 28,007 occasions the numbers on the dice added to 2.

You can copy these code snippets and run them in R, but you have to type them exactly as shown. There are only three exceptions:

- Don't type the prompt symbol, >.
- Don't type the continuation symbol, +.
- Where you put spaces or tabs isn't critical, as long as it isn't in the middle of a keyword. Pay attention to new lines, though.

Instructions to type code into the R console has the > symbol to the left:

```
> print("Hello world!")
```

If you type this into a console and press Enter, R responds with:

```
[1] "Hello world!"
```

For convenience, we collapse these two events into a single block, like this:

```
> print("Hello world!")
[1] "Hello world!"
```

Functions, arguments, and other R keywords appear in `monofont`. For example, to create a plot, you use the `plot()` function. Function names are followed by parentheses — for example, `plot()`. We don't add arguments to the function names mentioned in the text, unless it’s really important.

On some occasions we talk about menu commands, such as File⇒Save. This just means that you open the File menu and choose the Save option.

What You’re Not to Read

You can use this book however works best for you, but if you’re pressed for time (or just not interested in the nitty-gritty details), you can safely skip anything marked with a Technical Stuff icon. You also can skip sidebars (text in gray boxes); they contain interesting information, but nothing critical to your understanding of the subject at hand.

Foolish Assumptions

This book makes the following assumptions about you and your computer:

- **You know your way around a computer.** You know how to download and install software. You know how to find information on the Internet and you have Internet access.

- **You're not necessarily a programmer.** If you are a programmer, and you're used to coding in other languages, you may want to read the notes marked by the Technical Stuff icon — there, we fill you in on how R is similar to, or different from, other common languages.
- **You're not a statistician, but you understand the very basics of statistics.** *R For Dummies* isn't a statistics book, although we do show you how to do some basic statistics using R. If you want to understand the statistical stuff in more depth, we recommend *Statistics For Dummies,* 2nd Edition, by Deborah J. Rumsey, PhD (Wiley).
- **You want to explore new stuff.** You like to solve problems and aren't afraid of trying things out in the R console.

How This Book Is Organized

The book is organized in six parts. Here's what each of the six parts covers.

Part I: Getting Started with R Programming

In this part, you write your first script. You use the powerful concept of vectors to make simultaneous calculations on many variables at once. You work with the R workspace (in other words, how to create, modify, or remove variables). You find out how to save your work and retrieve and modify script files that you wrote in previous sessions. We also introduce some fundamentals of R (for example, how to install packages).

Part II: Getting Down to Work in R

In this part, we fill you in on the three R's: reading, 'riting, and 'rithmetic — in other words, working with text and numbers (and dates for good measure). You also get to use the very important data structures of *lists* and *data frames*.

Part III: Coding in R

R is a programming language, so you need to know how to write and understand functions. In this part, we show you how to do this, as well as how to control the logic flow of your scripts by making choices using `if` statements, as well as looping through your code to perform repetitive actions. We explain how to make sense of and deal with warnings and errors that you may experience in your code. Finally, we show you some tools to debug any issues that you may experience.

Part IV: Making the Data Talk

In this part, we introduce the different data structures that you can use in R, such as lists and data frames. You find out how to get your data in and out of R (for example, by reading data from files or the Clipboard). You also see how to interact with other applications, such as Microsoft Excel.

Then you discover how easy it is to do some advanced data reshaping and manipulation in R. We show you how to select a subset of your data and how to sort and order it. We explain how to merge different datasets based on columns they may have in common. Finally, we show you a very powerful generic strategy of splitting and combining data and applying functions over subsets of your data. When you understand this strategy, you can use it over and over again to do sophisticated data analyses in only a few small steps.

After reading this part, you'll know how to describe and summarize your variables and data using R. You'll be able to do some classical tests (for example, calculating a t-test). And you'll know how to use random numbers to simulate some distributions.

Finally, we show you some of the basics of using linear models (for example, linear regression and analysis of variance). We also show you how to use R to predict the values of new data using models that you've fitted to your data.

Part V: Working with Graphics

They say that a picture is worth a thousand words. This is certainly the case when you want to share your results with other people. In this part, you discover how to create basic and more sophisticated plots to visualize your data. We move on from bar charts and line charts,

and show you how to present cuts of your data using facets.

Part VI: The Part of Tens

In this part, we show you how to do ten things in R that you probably use Microsoft Excel for at the moment (for example, how to do the equivalent of pivot tables and lookup tables). We also give you ten tips for working with packages that are not part of base R.

Icons Used in This Book

As you read this book, you'll find little pictures in the margins. These pictures, or *icons,* mark certain types of text:

When you see the Tip icon, you can be sure to find a way to do something more easily or quickly.

You don't have to memorize this book, but the Remember icon points out some useful things that you really should remember. Usually this indicates a design pattern or idiom that you'll encounter in more than one chapter.

When you see the Warning icon, listen up. It points out something you definitely don't want to do. Although it's really unlikely that using R will cause something disastrous to happen, we use the Warning icon to alert you if something is bound to lead to confusion.

The Technical Stuff icon indicates technical information you can merrily skip over. We do our best to make this information as interesting and relevant as possible, but if you're short on time or you just want the information you absolutely *need* to know, you can move on by.

Beyond the Book

R For Dummies includes the following goodies online for easy download:

- **Cheat Sheet:** You can find the Cheat Sheet for this book here:

  www.dummies.com/cheatsheet/r

- **Extras:** We provide a few extra articles here:

  www.dummies.com/extras/r

- **Example code:** We provide the example code for the book here:

  www.dummies.com/extras/r

If we have updates to the content of the book, look here for it:

www.dummies.com/extras/r

Where to Go from Here

There's only one way to learn R: Use it! In this book, we try to make you familiar with the usage of R, but you'll have to sit down at your PC and start playing around with it yourself. Crack the book open so the pages don't flip by themselves, and start hitting the keyboard!

Part I

Getting Started with R Programming

Visit www.dummies.com for great Dummies content online.

In this part ...

- Introducing R programming concepts.
- Creating your first script.
- Making clear, legible code.
- Visit www.dummies.com for great Dummies content online.

Chapter 1

Introducing R: The Big Picture

In This Chapter

- Discovering the benefits of R
- Identifying some programming concepts that make R special

With an estimated worldwide user base of more than 2 million people, the R language has rapidly grown and extended since its origin as an academic demonstration language in the 1990s.

Some people would argue — and we think they're right — that R is much more than a statistical programming language. It's also

- A very powerful tool for all kinds of data processing and manipulation
- A community of programmers, users, academics, and practitioners
- A tool that makes all kinds of publication-quality graphics and data visualizations
- A collection of freely distributed add-on packages
- A versatile toolbox for extensive automation of your work

In this chapter, we fill you in on the benefits of R, as well as its unique features and quirks.

You can download R at www.r-project.org. This website also provides more information on R and links to the online manuals, mailing lists, conferences, and publications.

Tracing the history of R

Ross Ihaka and Robert Gentleman developed R as a free software environment for their teaching classes when they were colleagues at the University of Auckland in New Zealand. Because they were both familiar with S, a programming language for statistics, it seemed natural to use similar syntax in their own work. After Ihaka and Gentleman announced their software on the S-news mailing list, several people became interested and started to collaborate with them, notably Martin Mächler.

Currently, a group of 21 people has rights to modify the central archive of source code (http://www.r-project.org/contributors.html). This group is referred to as the R Core Team. In addition, many other people have contributed new code and bug fixes to the project.

Here are some milestone dates in the development of R:

- **Early 1990s:** The development of R began.
- **August 1993:** The software was announced on the S-news mailing list. Since then, a set of active R mailing lists has been created. The web page at www.r-project.org/mail.html provides descriptions of these lists and instructions for subscribing. (For more information, turn to "It provides an engaged community," later in this chapter.)
- **June 1995:** After some persuasive arguments by Martin Mächler (among others) to make the code available as "free software," the code was made available under the Free Software Foundation's GNU General Public License (GPL), Version 2.
- **Mid-1997:** The initial R Development Core Team was formed (although, at the time, it was simply known as the core group).
- **February 2000:** The first version of R, version 1.0.0, was released.
- **October 2004:** Release of R version 2.0.0.
- **April 2013:** Release of R version 3.0.0.
- **April 2015:** Release of R-3.2.0 (the version used in this book).

Ross Ihaka wrote a comprehensive overview of the development of R. The web page http://cran.r-project.org/doc/html/interface98-paper/paper.html provides a fascinating history.

Recognizing the Benefits of Using R

Of the many attractive benefits of R, a few stand out: It's actively maintained, it has good connectivity to various types of data and other systems, and it's versatile enough to solve problems in many domains. Possibly best of all, it's available for free, in more than one sense of the word.

It comes as free, open-source code

R is available under an open-source license, which means that anyone can download and modify the code. This freedom is often referred to as "free as in speech." R is also available free of charge — a second kind of freedom, sometimes referred to as "free as in beer." In practical terms, this means that you can download and use R free of charge.

As a result of this freedom, many excellent programmers have contributed improvements and fixes to the R code. For this reason, R is very stable and reliable.

Any freedom also has associated obligations. In the case of R, these obligations are described in the conditions of the license under which it is released: GNU General Public License (GPL), Version 2. The full text of the license is available at www.r-project.org/COPYING. It's important to stress that the GPL does not pertain to your usage of R. There are

no obligations for using the software — the obligations just apply to redistribution. In short, if you change *and* redistribute the R source code, you have to make those changes available for anybody else to use.

It runs anywhere

The R Core Team has put a lot of effort into making R available for different types of hardware and software. This means that R is available for Windows, Unix systems (such as Linux), and the Mac.

It supports extensions

R itself is a powerful language that performs a wide variety of functions, such as data manipulation, statistical modeling, and graphics. One really big advantage of R, however, is its extensibility. Developers can easily write their own software and distribute it in the form of add-on packages. Because of the relative ease of creating and using these packages, literally thousands of packages exist. In fact, many new (and not-so-new) statistical methods are published with an R package attached.

It provides an engaged community

The R user base keeps growing. Many people who use R eventually start helping new users and advocating the use of R in their workplaces and professional circles. Sometimes they also become active on

- The R mailing lists (http://www.r-project.org/mail.html
- Question-and-answer (Q&A) websites, such as
    - StackOverflow, a programming Q&A website (www.stackoverflow.com/questions/tagged/r)

- CrossValidated, a statistics Q&A website (http://stats.stackexchange.com/questions/tagged/r)

In addition to these mailing lists and Q&A websites, R users may

- Blog actively (www.r-bloggers.com).
- Participate in social networks such as Twitter (www.twitter.com/search/rstats).
- Attend regional and international R conferences.

See Chapter 11 for more information on R communities.

It connects with other languages

As more and more people moved to R for their analyses, they started trying to incorporate R in their previous workflows. This led to a whole set of packages for linking R to file systems, databases, and other applications. Many of these packages have since been incorporated into the base installation of R.

For example, the R package `foreign` (http://cran.r-project.org/web/packages/foreign/index.html) forms part of the *recommended* packages of R and enables you to read data from the statistical packages SPSS, SAS, Stata, and others (see Chapter 12).

Several add-on packages exist to connect R to database systems, such as

- `RODBC`, to read from databases using the Open Database Connectivity protocol (ODBC) (http://cran.r-project.org/web/packages/RODBC/index.html)
- `ROracle`, to read Oracle data bases (http://cran.r-project.org/web/packages/ROracle/index.html).

Initially, most of R was based on Fortran and C. Code from these two languages easily could be called from within R. As the community grew, C++, Java, Python, and other popular programming languages got more and more connected with R.

As more data analysts started using R, the developers of commercial data software no longer could ignore the new kid on the block. Many of the big commercial packages have add-ons to connect with R. Notably, both IBM's SPSS and SAS Institute's SAS allow you to move data and graphics between the two packages, and also call R functions directly from within these packages.

Other third-party developers also have contributed to better connectivity between different data analysis tools. For example, Statconn developed RExcel, an Excel add-on that allows users to work with R from within Excel (http://www.statconn.com/products.html).

Looking At Some of the Unique Features of R

R is more than just a domain-specific programming language aimed at data analysis. It has some unique features that make it very powerful, the most important one arguably being the notion of *vectors.* These vectors allow you to perform sometimes complex operations on a set of values in a single command.

Performing multiple calculations with vectors

R is a vector-based language. You can think of a *vector* as a row or column of numbers or text. The list of numbers `{1,2,3,4,5}`, for example, could be a vector. Unlike most other programming languages, R allows you to apply functions to the whole vector in a single operation without the need for an explicit loop.

It is time to illustrate vectors with some real R code. First, assign the values `1:5` to a vector called `x`:

```
> x <- 1:5
> x
[1] 1 2 3 4 5
```

Next, add the value `2` to each element in the vector `x`:

```
> x + 2
[1] 3 4 5 6 7
```

You can also add one vector to another. To add the values `6:10` element-wise to `x`, you do the following:

```
> x + 6:10
[1] 7 9 11 13 15
```

To do this in most other programming language would require an explicit loop to run through each value of `x`. However, R is designed to perform many operations in a single step. This functionality is one of the features that make R so useful — and powerful — for data analysis.

We introduce the concept of vectors in Chapter 2 and expand on vectors and vectorization in much more depth in Chapter 4.

Processing more than just statistics

R was developed by statisticians to make statistical data analysis easier. This heritage continues, making R a very powerful tool for performing virtually any statistical computation.

As R started to expand away from its origins in statistics, many people who would describe themselves as programmers rather than statisticians have become involved with R. The result is that R is now eminently suitable for a wide variety of nonstatistical tasks, including data processing, graphical visualization, and analysis of all sorts. R is being used in the fields of finance, natural language processing, genetics, biology, and market research, to name just a few.

R is *Turing complete,* which means that you can use R alone to program anything you want. (Not every task is easy to program in R, though.)

In this book, we assume that you want to find out about R programming, not statistics, although we provide an introduction to statistics with R in Part IV.

Running code without a compiler

R is an *interpreted language,* which means that — contrary to compiled languages like C and Java — you don't need a compiler to first create a program from your code before you can use it. R interprets the code you provide directly and converts it into lower-level calls to pre-compiled code/functions.

In practice, it means that you simply write your code and send it to R, and the code runs, which makes the development cycle easy. This ease of development comes at the cost of speed of code execution, however. The downside of an interpreted language is that the code usually runs slower than the equivalent compiled code.

If you have experience in other languages, be aware that R is *not* C or Java. Although you can use R as a procedural language such as C or an object-oriented language such as Java, R is mostly based on the functional programming paradigm. As we discuss later in this book, especially in Part III, this characteristic requires a bit of a different mindset. Forget what you know about other languages, and prepare for something completely different.

Chapter 2

Exploring R

In This Chapter

- Looking at your R editing options
- Starting R
- Writing your first R script
- Finding your way around the R environment

In order to start working in R, you need two things. First, you need a tool to easily write and edit code (an *editor).* You also need an *interface,* so you can send that code to R. Which tools you use depend to some extent on your operating system. The basic R install gives you these options:

- **Windows:** A basic user interface called **RGui.**
- **Mac OS X:** A basic user interface called **R.app.**
- **Linux:** There is no specific interface on Linux, but you can use any code editor (like Vim or Emacs) to edit your R code. R itself opens by default in a terminal window.

At a practical level, this difference between operating systems and interfaces doesn't matter very much. R is a programming language, and you can be sure that R interprets your code identically across operating systems.

Still, we want to show you how to use a standard R interface, so in this chapter we briefly illustrate how to

use R with the Windows **RGui.** Our advice also works on the Mac **R.app.**

Fortunately, there is an alternative, third-party interface called *RStudio* that provides a consistent user interface regardless of operating system. RStudio increasingly is the standard editing tool for R, so we also illustrate how to use RStudio.

In this chapter, after opening an R console, you flex your R muscles and write some scripts. You do some calculations, create some numeric and text objects, take a look at the built-in help, and save your work.

Working with a Code Editor

R is many things: a programming language, a statistical processing environment, a way to solve problems, and a collection of helpful tools to make your life easier. The one thing that R is *not* is an application, which means that you have the freedom of selecting your own editing tools to interact with R.

In this section we discuss the Windows R interface, RGui (short for *R graphical user interface*). This interface also includes a very basic editor for your code. Since this standard editor is so, well, basic, we also introduce you to RStudio. RStudio offers a richer editing environment than RGui and many handy shortcuts for common tasks in R.

Alternatives to the standard R editors

Among the many freedoms that R offers you is the freedom to choose your own code editor and development environment, so you don't have to use the standard R editors or RStudio.

These are powerful full-featured editors and development environments:

- **Eclipse StatET (**www.walware.de/goto/statet**):** Eclipse, another powerful integrated development environment, has an R add-in called StatET. If you've done software development on large projects, you may find Eclipse useful. Eclipse requires you to install Java on your computer.
- **Emacs Speaks Statistics (**http://ess.r-project.org**):** Emacs, a powerful text and code editor, is widely used in the Linux world and also is available for Windows. It has a statistics add-in called Emacs Speaks Statistics (ESS), which is famous for having keyboard shortcuts for just about everything you could possibly do and for its very loyal fan base. If you're a programmer coming from the Linux world, this editor may be a good choice for you.
- **Tinn-R (**http://nbcgib.uesc.br/lec/software/editores/tinn-r/en**):** This editor, developed specifically for working with R, is available only for Windows. It has some nice features for setting up collections of R scripts in projects. Tinn-R is easier to install and use than either Eclipse or Emacs, but it isn't as fully featured.

A couple of interfaces are designed as tools for special purposes:

- **Rcommander (**http://www.rcommander.com/**):** Rcommander provides a simple GUI for data analysis in R and contains a variety of plugins for different tasks.
- **Rattle (**http://rattle.togaware.com/**):** Rattle is a GUI designed for typical data mining tasks.

Exploring RGui

As part of the process of downloading and installing R, you get the standard graphical user interface (GUI), called *RGui.* RGui gives you some tools to manage your R environment — most important, a console window. The console is where you type instructions and generally get R to do useful things for you.

Seeing the naked R console

The standard installation process creates useful menu shortcuts (although this may not be true if you use Linux, because there is no standard GUI interface for Linux). In the menu system, look for a folder called R, and then find an icon called R followed by a version number (for example, R 3.2.0, as shown in Figure 2-1).

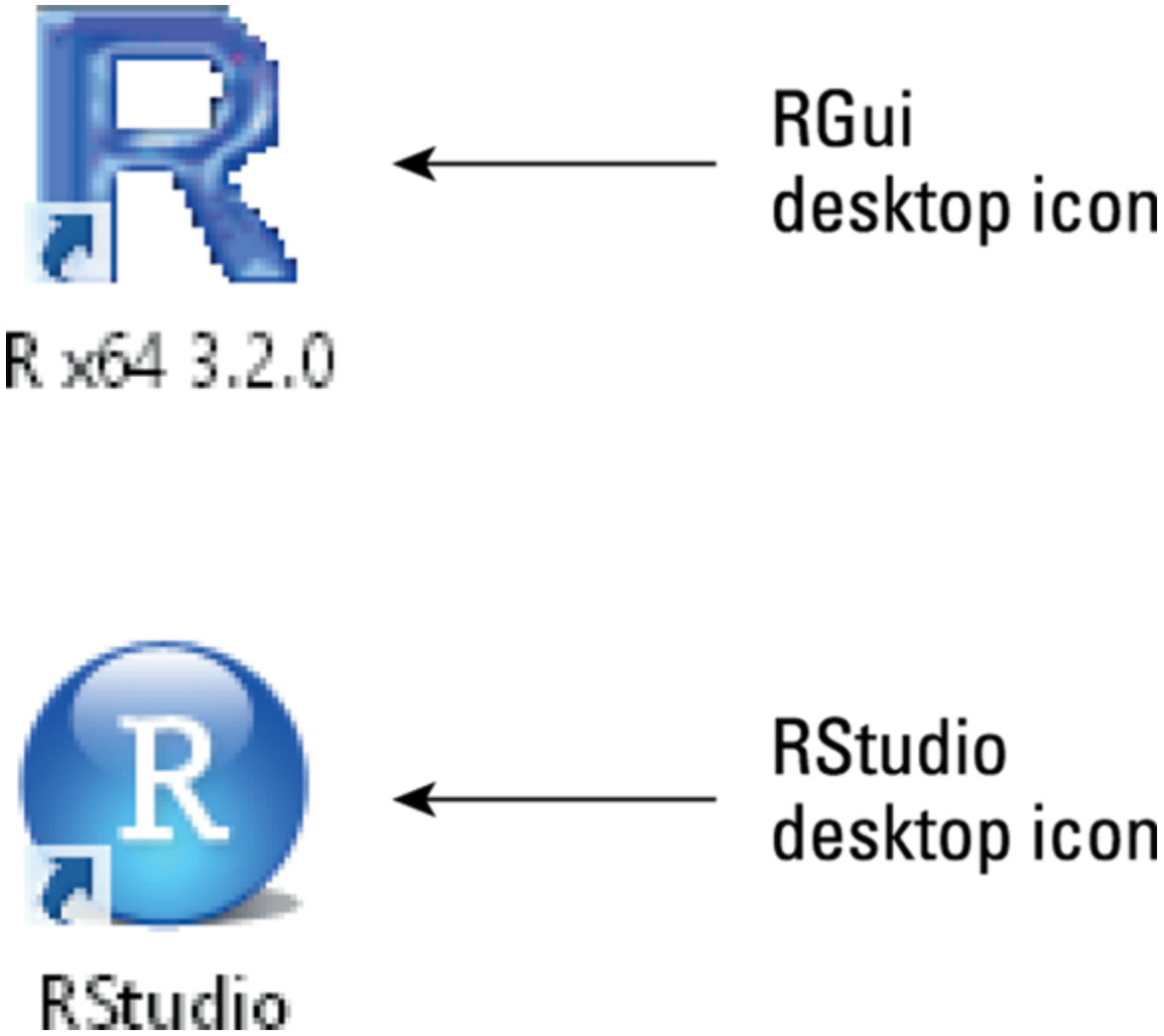

Figure 2-1: Shortcut icons for RGui (R x64) and RStudio.

When you open RGui for the first time, you see the R *Console* screen (shown in Figure 2-2), which lists some basic information such as your version of R and the licensing conditions.

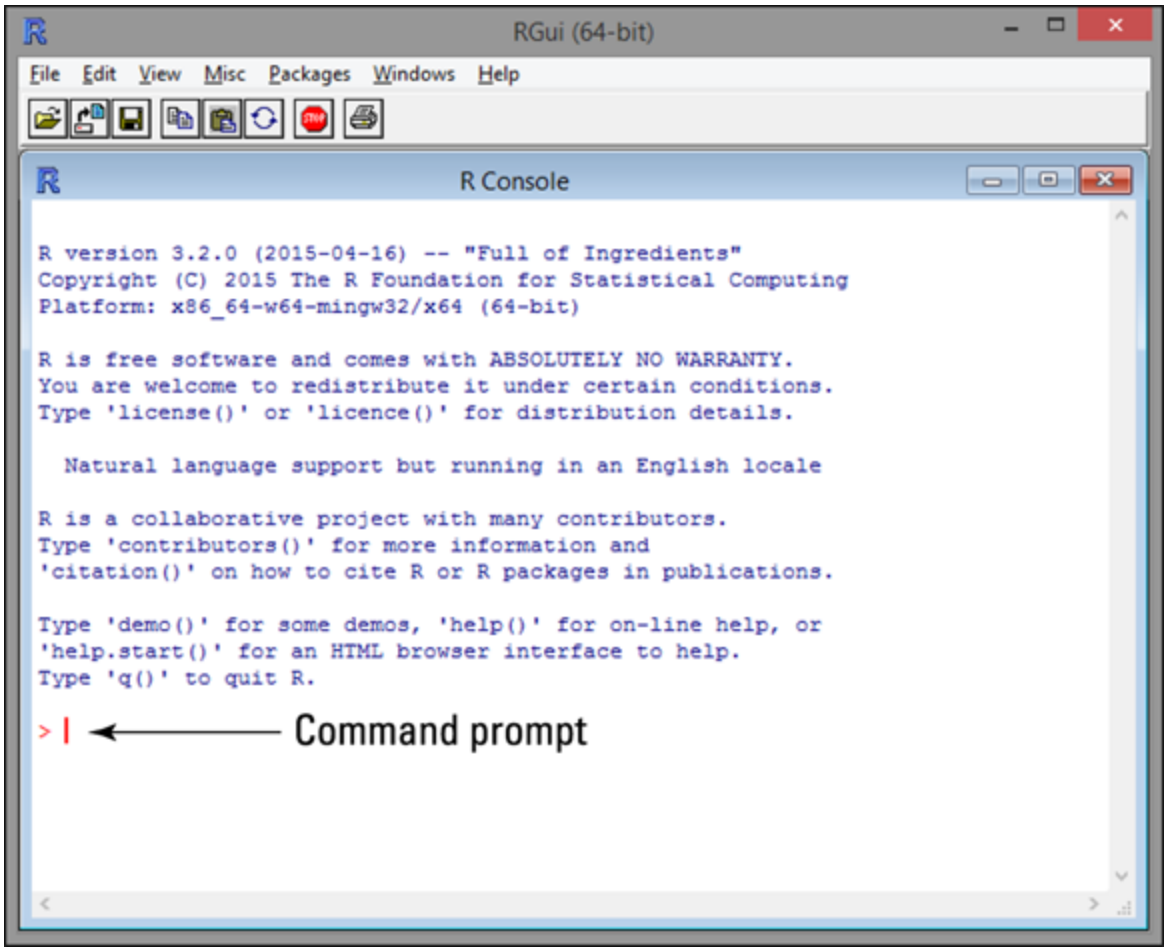

Figure 2-2: A brand-new session in RGui.

Below all this information is the R *prompt,* denoted by a > symbol. The prompt indicates where you type your commands to R; you see a blinking cursor to the right of the prompt.

We explore the R console in more depth in "Navigating the Environment," later in this chapter.

Issuing a simple command

Use the console to issue a very simple command to R. Type the following to calculate the sum of some numbers, directly after the prompt:

```
> 24 + 7 + 11
```

R responds immediately to your command, calculates and displays the total in the console:

```
> 24 + 7 + 11
[1] 42
```

The answer is 42. R gives you one other piece of information: The `[1]` preceding 42 indicates that the value 42 is the first element in your answer. It is, in fact, the only element in your answer! One of the clever things about R is that it can deal with calculating many values at the same time, which is called *vector operations.* We talk about vectors later in this chapter — for now, all you need to know is that R can handle more than one value at a time.

Closing the console

To quit your R session, type the following code in the console, after the command prompt (>):

```
> quit()
```

R asks you a question to make sure that you meant to quit, as shown in Figure 2-3. Click No, because you have nothing to save. This action closes your R session (as well as RGui, if you've been using RGui as your code editor). In fact, saving a workspace image rarely is useful.

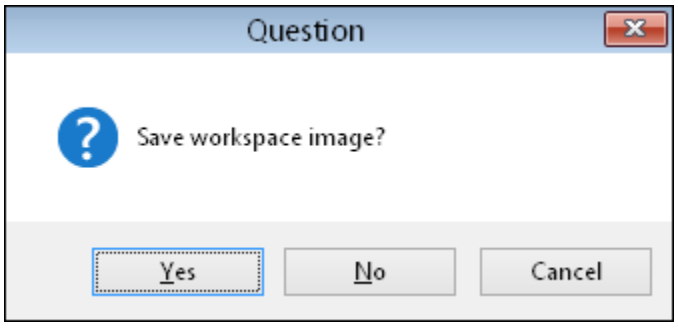

Figure 2-3: R asks you a simple question.

Dressing up with RStudio

RStudio is a code editor and development environment with some very nice features that make code development in R easy and fun:

- Code highlighting that gives different colors to keywords and variables, making it easier to read
- Automatic bracket and parenthesis matching
- Code completion, so you don't have to type out all commands in full
- Easy access to R Help, with some nice features for exploring functions and parameters of functions
- Easy exploration of variables and values

Because RStudio is available free of charge for Linux, Windows, and Apple OS X, we think it's a good option to use with R. In fact, we like RStudio so much that we use it to illustrate the examples in this book. Throughout the book, you find some tips and tricks on how things can be done in RStudio. If you decide to use a different code editor, you can still use all the code examples and you'll get identical results.

To open RStudio, click the RStudio icon in your menu system or on your desktop. (You can find installation instructions in this book's appendix.)

Once RStudio starts, choose File⇒New⇒R Script to open a new script file.

Your screen should look like Figure 2-4. You have four work areas (also called panes):

- **Source:** The top-left corner of the screen contains a text editor that lets you work with source script files. Here, you can enter multiple lines of code, save your script file to disk, and perform other tasks on your

script. This code editor works a bit like every other text editor you've ever seen, but it's smart. It recognizes and highlights various elements of your code, for example (using different colors for different elements), and it also helps you find matching brackets in your scripts.

- **Console:** In the bottom-left corner, you find the console. The console in RStudio can be used in the same way as the console in RGui (refer to "Seeing the naked R console," earlier in this chapter). This is where you do all the interactive work with R.
- **Environment and History:** The top-right corner is a handy overview of your environment, where you can inspect the variables you created in your session, as well as their values. (We discuss the environment in more detail later in this chapter.) This is also the area where you can see a history of the commands you've issued in R.
- **Files, plots, package, help, and viewer:** In the bottom-right corner, you have access to several tools:
  - **Files:** This is where you can browse the folders and files on your computer.
  - **Plots:** This is where R displays your plots (charts or graphs). We discuss plots in Part V.
  - **Packages:** You can view a list of all installed packages.

    

    A *package* is a self-contained set of code that adds functionality to R, similar to the way that add-ins add functionality to Microsoft Excel.
  - **Help:** This is where you can browse R's built-in Help system.