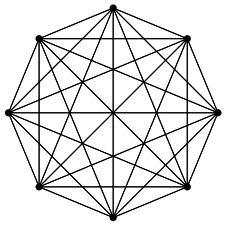
Chapter 1

Introducing R



R is a programming language that has steadily grown in popularity over the past few decades. Regardless of whether you work in industry, academics, or government, or if you just use the language for personal use, you will always have access to the R language. So why use R? There are at least eight answers to this question.

1. R is open source software. It can be downloaded at no cost onto a desktop or a laptop computer.

2. R is capable of performing numerical calculations on scalars, vectors, and matrices. It can be used as a high-powered calculator.

3. R has built-in functions for performing probability and statistical calculations. R can perform simulation experiments using these functions.

4. R has extensive graphical capabilities. R can generate production-level graphics using built-in graphical functions. These flexible functions can generate customized graphics.

5. R is a programming language. R allows the user to execute several types of loops for iteration and supports conditional execution of code.

6. R is reproducible. R commands posted on the internet, for example, can simply be copied, pasted, and executed in your current R session.

7. R has a vast array of support resources, including textbooks, on-line tutorials, on-line answers to specific questions, an academic journal, and the annual international *UseR!* conference.

8. R has thousands of contributors who have written R code contained in *packages*, which continue to extend the language in the same way that apps have transformed handheld devices.

The title of this book, *Learning Base R*, emphasizes that the portion of R introduced here is the "base" language, which is what you encounter when you initially enter an R session. The final chapter considers *packages*, which substantially extend the capability of the base language.

The R language, originally known as S, was developed in 1976 at Bell Labs by John Chambers and his colleagues. S-Plus is the current commercial version of S. In 1993, the free, open source R language was first developed by Ross Ihaka and Robert Gentleman (notice the common first initial) from the University of Auckland. R and S have a very large overlap of capabilities. If you know one, you essentially know the other. The differences are minor. One advantage to R over S, however,

is the vast number of packages that have been written in R that are capable of extending the base language. These packages can be quite useful in certain applications.

Many R users prefer to use R in an *integrated development environment*, often abbreviated IDE. One popular IDE for R is RStudio, which makes the process of writing an R program more convenient than working in the native R language. Several panes open when you enter RStudio. The lower-left pane usually corresponds to a native R session. This book applies equally well to native R and RStudio. If you are new to R and unsure which platform to use, try RStudio for its convenience in developing R programs.

The orientation for R is as follows. R is a vector-based language that uses vectors as its primary data structure. R has a command-line orientation rather than a Graphical User Interface (GUI) menu orientation. Although this might seem a bit antiquated at first, the size and capability of the language force this orientation. The focus here will be on the syntax: the rules for issuing R commands. There are plenty of other R books on the market that are much more encyclopedic in nature or cover specific applications. This book is designed to be a quick introduction to the language for R novices. More advanced books on R will be much easier to read once you master the basics presented here.

You will notice that R tends to favor short, abbreviated variable and function names, which is helpful on your fingers in terms of saving keystrokes. This is a heritage from the C language and the Unix environment that was present at Bell Labs in the 1970s.

The R language can be located by doing a web search on the letter R in a browser. Alternatively, you can go directly to the website `http://www.r-project.org`. Once you get to the website you should pick a nearby mirror site and install the binary version. You will need to choose an appropriate platform: Windows, Mac, or Linux. You might also need to choose between the 32-bit and 64-bit version of R. For your first installation, it is probably best to select the default location for installing R. Installing R is similar to installing any other type of software. A license agreement must be accepted and you will want to install the most recent base version of the software. In addition to the version number, there is an associated release name, such as "Lost Library Book," "You Stupid Darkness," "Bug in Your Hair," or "Great Pumpkin." The source of these curious release names is unclear, but they seem to reflect the season of the release, and many are taken from Charles Schulz's *Peanuts* comic strips and films. R is a fairly large language, so installing it typically takes a few minutes. On most platforms, an icon with the letter R will appear on your desktop after downloading and installation. To launch R or RStudio, double-click the appropriate icon on your desktop. This will bring up a window which contains a greater-than prompt, which looks like this:

```
>
```

The prompt indicates that R is waiting for an R command from you. The remainder of this book describes what you can type after the prompt and the associated results. If you type any valid R command at this point and press the return key, R will display the output. If a command is too long for a single line, it is fine to press the return key and continue the command over several lines. A + prompt automatically appears to remind you that you are completing a command from the previous line. If a command requires significant processing time, there will be a time delay for processing before a new prompt appears. This orientation of having a prompt which awaits an R command from you is known as the *command line interface*. The output from an R command might vary slightly depending on the platform (Windows, Mac, or Linux) on which R is running.

When your R session is finished, you can quit R by typing `q()`, for quit, at the prompt. A dialog box will open that asks: `Save workspace image? [y/n/c]`. This determines whether R saves the objects that you have created during the R session for a future R session. After responding to that question with `y` or `n`, for *yes* or *no*, your R session is completed. Pressing `c` for *cancel* allows the R session to continue.

Experienced programmers who are used to working with compiled languages may find R's orientation foreign. The practice of submitting a command, viewing the results, then perhaps submitting a subsequent command based on the results is not the way compiled languages are generally used. This orientation is largely a result of the roots of R in exploratory data analysis (sometimes abbreviated EDA). One preliminary look at the data often guides subsequent analysis techniques. It is easy to execute a series of R commands that are stored in an external file if the exploratory data analysis approach is not appropriate in a particular setting. Also, R stores its matrices in a column-major orientation, as opposed to the more standard row-major orientation. This is also an artifact of R's roots in exploratory data analysis and the associated data structures required to efficiently store data sets.

R is an interpreted rather than a compiled language, which means that each R command must be interpreted and then executed. Therefore, run times for R code will be a bit slower than comparable code in a compiled language. Some speedup is possible, however, with vector-based programming, which will be described later in the text.

Comments are useful for documenting the purpose of a particular R command or a group of R commands either for yourself or for someone else who might use your code. A comment consists of text that is ignored by R as it processes your R input command. When R encounters the # symbol, it ignores all subsequent input on that line.

As you begin to wade into R, there are several tricks that can make your R session more efficient. Here are five such tricks.

1. You can browse through previous R commands using the up arrow. If you encounter a command that you would like to alter, you can use the destructive backspace, or the nondestructive backspace (the left arrow) to edit the R command. Pressing the enter key executes the modified R command even if the cursor is not at the end of the line.

2. Tab completion is another way to save keystrokes. For example, if you type

   ```
   > VA
   ```

   at the R prompt and then press the tab key, R recognizes that `VADeaths` is the only object that begins with the letters `VA`, so it completes the name of the object.

3. Sometimes your R session can get cluttered. Typing control-ℓ clears the screen in both native R and in the console pane of RStudio.

4. Several R commands can be executed at once. In RStudio, the commands are keyed into the code development pane. Pressing the *Run* button executes the current line or the highlighted lines. In native R, enter R by double-clicking on the R icon. Using the `File` drop-down menu, click on `New Document`, which opens a window for the R commands. After the R commands are keyed in, command-enter executes the current line or the highlighted lines.

5. R code that you encounter on the internet can be copied and pasted into your R session to be executed. As with anything taken from the internet, proceed with caution.

The R language does not have a natural way to sequence the introduction of the various topics. I have decided to sequence the subsequent chapters in the following fashion.

- Chapters 2 and 3 introduce elementary arithmetic operations and named storage in R.
- Chapters 4, 5, and 6 introduce three elementary data structures: vectors, matrices, and arrays. In these early chapters, these data structures are filled with numeric elements.

- Chapters 7 and 8 introduce built-in and user-written functions.
- Chapter 9 introduces some useful utilities.
- Chapters 10, 11, and 12 introduce three other types of elements that can be stored in a data structure: complex numbers, character strings, and logical elements.
- Chapters 13 and 14 introduce methods for comparing elements with relational operators and coercing elements to a particular data type.
- Chapters 15 and 16 introduce two advanced data structures: lists and data frames.
- Chapter 17 surveys some data sets that are built into R.
- Chapter 18 introduces input and output functions that interface R with the outside world.
- Chapter 19 introduces functions that are useful in probability calculations.
- Chapters 20 and 21 introduce R's graphical capabilities.
- Chapters 22, 23, and 24 introduce R's programming capabilities.
- Chapter 25 shows how to conduct simulations in R.
- Chapter 26 surveys R's capability for performing statistical inference.
- Chapter 27 introduces linear algebra functions that are built into R.
- Chapter 28 shows how to extend R's base capability by using packages.

You will find R to be a very efficient language for performing both simple and complex calculations. R can be used for analyzing data, performing simulations, generating graphics, etc. The value of a language like R was recognized long ago.

> Civilization advances by extending the number of important operations which we can perform without thinking about them. —Alfred North Whitehead (1861–1947)

The next chapter introduces R in its simplest form—as a calculator.

Exercises

1.1 Download R from the website `http://www.r-project.org` onto your desktop or laptop computer. Optionally, download RStudio from the website `https://www.rstudio.com`. If you decide to use RStudio instead of native R, you must also download R.

1.2 The Comprehensive R Archive Network (CRAN) website `http://cran.r-project.org` contains links to various sites concerning the R language. Browse some of the online documentation for R and write a paragraph describing a site that you found interesting.

1.3 Launch an R session and practice with tab completion.

1.4 Launch an R session and practice using the up arrow to repeat previous commands.