1

Getting started

Welcome to the world of statistical programming. We will start in this chapter by giving you an idea of what statistical programming is all about. We will also tell you what to expect as you proceed through the rest of the book. The chapter will finish with some instructions about how to download and install R, the software package and language on which we base our programming examples, and RStudio, an "integrated development environment" (or "IDE") for R.

1.1 What is statistical programming?

Computer programming involves controlling computers, telling them what calculations to do, what to display, etc. Statistical programming is harder to define. One definition might be that it's the kind of computer programming statisticians do—but statisticians do all sorts of programming. Another would be that it's the kind of programming one does when one is doing statistics: but again, statistics involves a wide variety of computing tasks.

For example, statisticians are concerned with collecting and analyzing data, and some statisticians would be involved in setting up connections between computers and laboratory instruments: but we would not call that statistical programming. Statisticians often oversee data entry from questionnaires, and may set up programs to aid in detecting data entry errors. That *is* statistical programming, but it is quite specialized, and beyond the scope of this book.

Statistical programming involves doing computations to aid in statistical analysis. For example, data must be summarized and displayed. Models must be fit to data, and the results displayed. These tasks can be done in a number of different computer applications: Microsoft Excel, SAS, SPSS, R, Stata, etc. Using these applications is certainly statistical computing, and usually involves statistical programming, but it is not the focus of this book. In this book our aim is to provide a foundation for an understanding of how those applications work: what are the calculations they do, and how could you do them yourself.

Since graphs play an important role in statistical analysis, drawing graphics of one, two or higher dimensional data is an aspect of statistical programming.

An important part of statistical programming is stochastic simulation. Digital computers are naturally very good at exact, reproducible computations, but the real world is full of randomness. In stochastic simulation we program a computer to act as though it is producing random results, even though if we knew enough, the results would be exactly predictable.

Statistical programming is closely related to other forms of numerical programming. It involves optimization, and approximation of mathematical functions. Computational linear algebra plays a central role. There is less emphasis on differential equations than in physics or applied mathematics (though this is slowly changing). We tend to place more of an emphasis on the results and less on the analysis of the algorithms than in computer science.

1.2 Outline of this book

This book is an introduction to statistical programming. It contains a lot of specific advice about the hows and whys of the subject. We will start with basic programming: how to tell a computer what to do. We do this using the open source R statistical package, so we will teach you R, but we will try not to *just* teach you R. We will emphasize those things that are common to many computing platforms.

Statisticians need to display data. We will show you how to construct statistical graphics. In doing this, we will learn a little bit about human vision, and how it motivates our choice of display.

In our introduction to programming, we will show how to control the flow of execution of a program. For example, we might wish to do repeated calculations as long as the input consists of positive integers, but then stop when an input value hits 0. Programming a computer requires basic logic, and we will touch on Boolean algebra, a formal way to manipulate logical statements. The best programs are thought through carefully *before* being implemented, and we will discuss how to break down complex problems into simple parts. When we are discussing programming, we will spend quite a lot of time discussing how to *get it right*: how to be sure that the computer program is calculating what you want it to calculate. We will talk about the importance of consistency and clarity in complex projects, using the example of the *tidyverse* project for data science.

One distinguishing characteristic of statistical programming is that it is concerned with randomness: random errors in data, and models that include stochastic components. We will discuss methods for simulating random values with specified characteristics, and show how random simulations are useful in a variety of problems.

Many statistical procedures are based on linear models. While discussion of linear regression and other linear models is beyond the scope of this book, we do discuss some of the background linear algebra, and

how the computations it involves can be carried out. We also discuss the general problem of numerical optimization: finding the values which make a function as large or as small as possible.

Each chapter has a number of exercises which are at varying degrees of difficulty. Solutions to selected exercises can be found on the web at `www.statprogr.science`.

1.3 The R package

This book uses R, which is an open source package for statistical computing. "Open source" has a number of different meanings; here the important one is that R is freely available, and its users are free to see how it is written, and to improve it. R is based on the computer language S, developed by John Chambers and others at Bell Laboratories in 1976. In 1993 Robert Gentleman and Ross Ihaka at the University of Auckland wanted to experiment with the language, so they developed an implementation, and named it R. They made it open source in 1995, and thousands of people around the world have contributed to its development.

1.4 Why use a command line?

The R system is mainly command-driven, with the user typing in text and asking R to execute it. Nowadays most programs use interactive graphical user interfaces (menus, touchscreens, etc.) instead. So why did we choose such an old-fashioned way of doing things?

Menu-based interfaces are very convenient when applied to a limited set of commands, from a few to one or two hundred. However, a command-line interface is open ended. As we will show in this book, if you want to program a computer to do something that no one has done before, you can easily do it by breaking down the task into the parts that make it up, and then building up a program to carry it out. This may be possible in some menu-driven interfaces, but it is much easier in a command-driven interface.

Moreover, learning how to use one command-line interface will give you skills that carry over to others, and may even give you some insight into how a menu-driven interface is implemented. As statisticians, it is our belief that your goal should be understanding, and learning how to program at a command line will give you that at a fundamental level. Learning to use a menu-based program makes you dependent on the particular organization of that program.

There is no question that command-line interfaces require greater knowledge on the part of the user—you need to remember what to type to achieve a particular outcome. Fortunately, there is help. We recommend that you use the RStudio integrated development environment (IDE). IDEs were first developed in the 1970s to help programmers: they allow you to

edit your program, to search for help, and to run it; when your first attempt doesn't work, they offer support for diagnosing and fixing errors. RStudio is an IDE for R programming, first released in 2011. It is produced by a Public Benefit Corporation[1] named RStudio, and is available for free use.

[1] Public Benefit Corporations are for-profit corporations whose corporate decisions must balance the interests of community, customers, employees, and shareholders.

1.5 Font conventions

This book describes how to do computations in R. As we will see in the next chapter, this requires that the user types input, and R responds with text or graphs as output. To indicate the difference, we have typeset the user input and R output in a tinted box. The output is prefixed with ##. For example

```
This was typed by the user
```

```
## This is a response from R
```

In most cases other than this one and certain exercises, we will show the actual response from R corresponding to the preceding input.[2]

[2] We have used the `knitr` package so that R itself is computing the output. The computations in the text were done with R version 4.0.2 (2020-06-22).

There are also situations where the code is purely illustrative and is not meant to be executed. (Many of those are not correct R code at all; others illustrate the syntax of R code in a general way.) In these situations we have typeset the code examples in an upright typewriter font. For example,

```
f( some arguments )
```

1.6 Installation of R and RStudio

R can be downloaded from `https://cloud.r-project.org`. Most users should download and install a *binary version*. This is a version that has been translated (by *compilers*) into machine language for execution on a particular type of computer with a particular operating system. R is designed to be very *portable*: it will run on Microsoft Windows, Linux, Solaris, macOS, and other operating systems, but different binary versions are required for each. In this book most of what we do would be the same on any system, but when we write system-specific instructions, we will assume that readers are using Microsoft Windows.

Installation on Microsoft Windows is straightforward. A binary version is available for Windows Vista or above from the web page `https://cloud.r-project.org/bin/windows/base`. Download the "setup program," a file with a name like `R-4.0.2-win.exe`. Clicking on this file will start an almost automatic installation of the R system. Though it is possible to customize the installation, the default responses will lead to a satisfactory installation in most situations, particularly for beginning users.

One of the default settings of the installation procedure is to create an R icon on your computer's desktop.

You should also install RStudio, after you have installed R. As with R, there are separate versions for different computing platforms, but they all look and act similarly. You should download the "Open Source Edition" of "RStudio Desktop" from `www.rstudio.com`, and follow the instructions to install it on your computer.

1.7 Getting started in RStudio

Once you have installed R and RStudio, you will be ready to start statistical programming. We'll start with a quick tour of RStudio, and introduce more detail in later chapters.

When you are working in RStudio, you'll see a display something like Figure 1.1. (The first time you start it, you won't see all the content that is in the figure. After working with it for a while, you may have customized it to look quite different!) The display includes four *panes*. The top left pane is the *Source Pane*, or editor. You will type your program (or other document) there. You can have several open files in this pane; the tabs allow you to choose which is active. The bottom left pane is called the *Console Pane*.

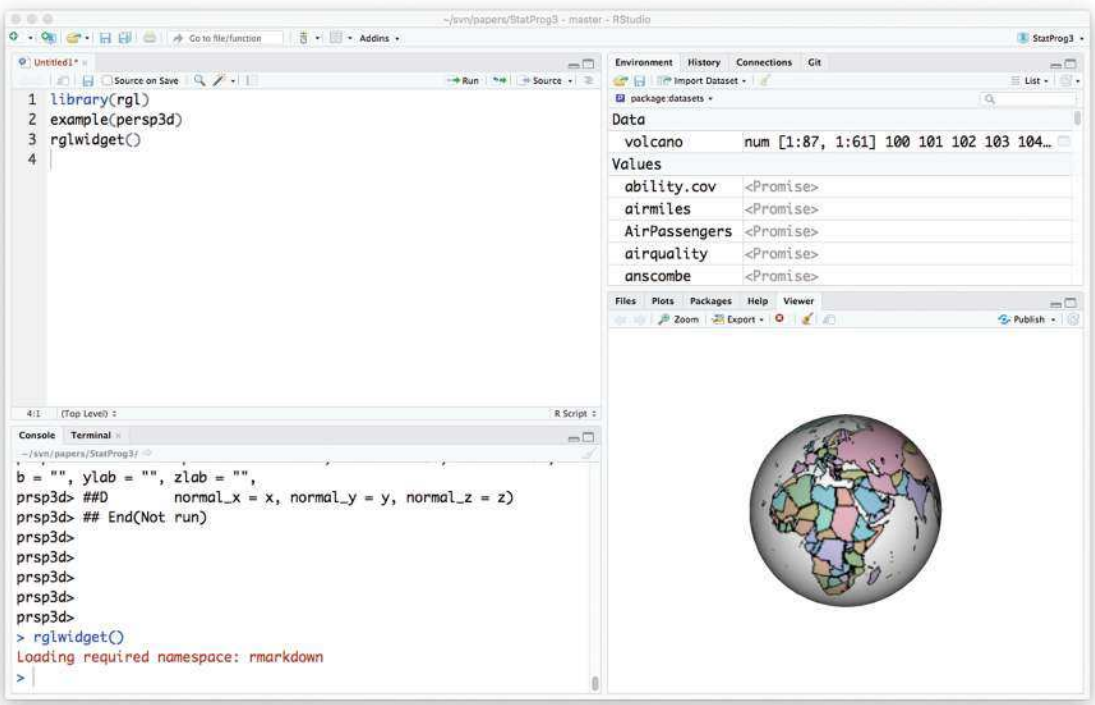

Fig. 1.1 A typical RStudio display.

This is where you communicate with R. You can type directly into this pane, but it is usually better to work within the editor pane, because that way you can easily correct mistakes and try again.

The two right-hand panes contain a variety of *tabs*. In the figure, the top pane is showing the *Environment Pane* (i.e. the workspace), and the bottom pane is showing a plot; we'll discuss these and the other tabs in later chapters. For now, you just need to know the following points:

- You should do most of your work in the editor, but you can occasionally type in the console.
- The console pane displays what R is doing.
- All of the panes can be resized and repositioned, so sometimes it may appear that you've lost one, but there's no need to worry: just find the header of the pane and click there with your mouse, and the pane will reappear. If the pane is there but the content isn't what you want, try clicking on the tabs at the top.

1.8 Going further

This book introduces statistical programming with R, but doesn't come close to covering everything. Here are some further resources.

- There are many textbooks that will teach you more about statistics. We recommend *Data Analysis and Graphics Using R: An Example-Based Approach* by Maindonald and Braun and *Introductory Statistics with R* by Dalgaard for an introductory level presentation, and the classic *Modern Applied Statistics with S* by Venables and Ripley for more advanced material. *R for Data Science* by Grolemund and Wickham describes the "tidy" approach to data science that we will introduce in Chapter 5 and *Advanced R* by Wickham gives more detail about programming in R.
- There are many tools that use R in preparing printed documents. We particularly like `knitr`, which you can read about online at `https://yihui.name/knitr` or in the book *Dynamic Documents with R and knitr* by Xie. It provides a very rich system; for a simple subset (useful to write your assignments for class!), we describe R Markdown in Section 2.7.3. See `https://rmarkdown.rstudio.com/` for more details on this subset.
- R can also be used to prepare interactive web pages. The Shiny system displays output from R based on prepared scripts that are controlled in a browser. The user doesn't need to install R, but he or she can see R output. You can see an example and read more about Shiny at `https://shiny.rstudio.com`.