1

Getting started

Welcome to the world of statistical programming. This book contains a lot of specific advice about the hows and whys of the subject. We 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, S-PLUS, 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. 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.

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 Boston company named RStudio, and is available for free use.

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 gray 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.[1]

[1] We have used the knitr package so that R itself is computing the output. The computations in the text were done with R version 3.2.2 (2015-08-14).

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 )
```

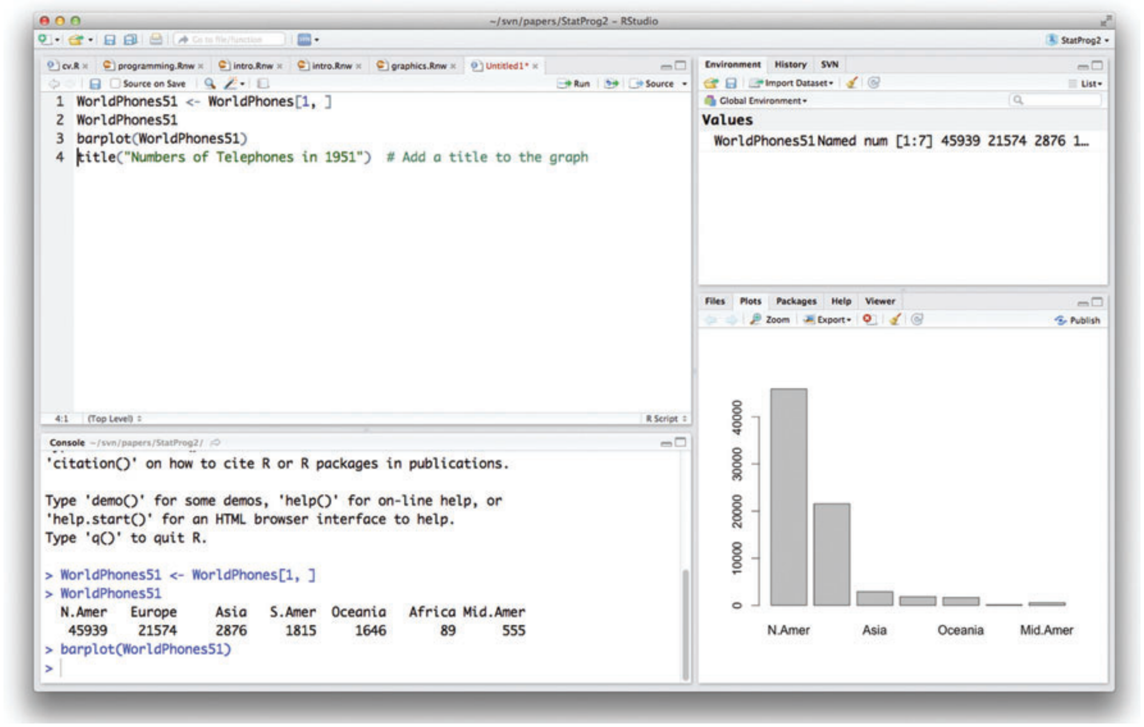
1.6 Installation of R and RStudio

R can be downloaded from `http://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, Mac OSX, 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 `http://cloud.r-project.org/bin/windows/base`. Download the "setup program," a file with a name like `R-3.2.5-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.



```
WorldPhones51 <- WorldPhones[1, ]
WorldPhones51
barplot(WorldPhones51)
title("Numbers of Telephones in 1951")  # Add a title to the graph
```

```
WorldPhones51 Named num [1:7] 45939 21574 2876 1…
```

```
'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.

> WorldPhones51 <- WorldPhones[1, ]
> WorldPhones51
 N.Amer  Europe    Asia  S.Amer Oceania  Africa Mid.Amer
  45939   21574    2876    1815    1646      89      555
> barplot(WorldPhones51)
>
```

Fig. 1.1 A typical RStudio display.

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.) The display includes four *panes*. The top left pane is the *Source Pane*, or editor. You will type your program (or other document) there. The bottom left pane is called the *Console Pane*. 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 *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. *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 `http://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!), take a look at R Markdown (`http://rmarkdown.rstudio.com/`).
- 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 `http://shiny.rstudio.com`.

2

Introduction to the R language

Having installed the R and RStudio systems, you are now ready to begin to learn the art of statistical programming. The first step is to learn the *syntax* of the language that you will be programming in; you need to know the rules of the language. This chapter will give you an introduction to the syntax of R. Most of what we discuss here relates to what you would type into the R console or into the RStudio script window.

2.1 First steps

Having opened R or RStudio, you may begin entering and executing commands, usually interactively. Normally, you will use the Source Pane to type in your commands, but you may occasionally use the Console Pane directly. The greater-than sign (>) is the prompt symbol which appears in the Console Pane.

2.1.1 R can be used as a calculator

Anything that can be computed on a pocket calculator can be computed at the R prompt. The basic operations are + (add), - (subtract), * (multiply), and / (divide). For example, try

```
5504982/131071
```

Upon pressing the Enter key (or CTRL-Enter), the result of the above division operation, 42, appears in the Console Pane, preceded by the command you executed, and prefixed by the number 1 in square brackets:

```
5504982/131071

## [1] 42
```

The [1] indicates that this is the first (and in this case only) result from the command. Many commands return multiple values, and each line of results will be labeled to aid the user in deciphering the output. For example, the sequence of integers from 17 to 58 may be displayed as follows:

```
17:58

##  [1] 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
## [23] 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58
```

The first line starts with the first return value, so it is labeled `[1]`; the second line starts with the 23rd, so it is labeled `[23]`.

Everything that you type after a # sign is assumed to be a comment and is ignored by R.

```
5:(2*3 + 10)    # the result is  the same as 5:16

##  [1]  5  6  7  8  9 10 11 12 13 14 15 16

(7:10) + pi       # pi is a stored constant

## [1] 10.14159 11.14159 12.14159 13.14159
```

Note the use of parentheses in the examples above. Parentheses are used to ensure that the operations (in this case, `:`, `*`, and +) are carried out in the order that we desire. In the first case, parentheses were necessary to obtain the result we wanted to see. The following shows what happens when the parentheses are omitted:

```
5:2*3 + 10

## [1] 25 22 19 16
```

If you are surprised by this result, it would be a good exercise to break the calculation down into the three separate operations in order to determine exactly what R is doing.

The parentheses were not required in `(7:10) + pi`. We used them anyway, for two reasons. First, they can help others read and understand the code more quickly. Second, although R follows strict and consistent rules regarding order of operations, we believe it is too easy for a user to forget one or more of these rules. Therefore, we recommend using parentheses whenever you are unsure (or even in cases where you think you may be right).

R can also be used to compute powers with the ^ operator. For example,

```
3^4

## [1] 81
```

Modular arithmetic is also available. For example, you can compute the remainder after division of 31 by 7, i.e. 31 (mod 7):

```
31 %% 7

## [1] 3
```

and the integer part of a fraction as

```
31 %/% 7

## [1] 4
```

We can confirm that 31 is the sum of its remainder plus seven times the integer part of the fraction:

```
7*4 + 3

## [1] 31
```

2.1.2 Named storage

R has a workspace known as the *global environment* that can be used to store the results of calculations, and many other types of objects. For a first example, suppose we would like to store the result of the calculation `1.0025^30` for future use. (This might arise from a compound interest calculation based on an interest rate of 0.25% per year and a 30-year term.) We will assign this value to an object called `interest.30`. To do this, we type

```
interest.30 <- 1.0025^30
```

We tell R to make the assignment using an arrow that points to the left, created with the less-than sign (`<`) and the hyphen (`-`). R also supports using the equals sign (`=`) in place of the arrow in most circumstances, but we recommend using the arrow, as it makes clear that we are requesting an *action* (i.e. an assignment), rather than stating a *relation* (i.e. that `interest.30` is equal to `1.0025^30`), or making a permanent definition. Note that when you run this statement, no output appears: R has done what we asked, and is waiting for us to ask for something else.

You can see the results of this assignment by typing the name of our new object at the prompt:

```
interest.30

## [1] 1.077783
```

Think of this as just another calculation: R is calculating the result of the expression `interest.30`, and printing it. You can also use `interest.30` in further calculations if you wish. For example, you can calculate the bank balance after 30 years at 0.25% annual interest, if you start with an initial balance of $3000:

```
initialBalance <- 3000
finalBalance <- initialBalance * interest.30
finalBalance

## [1] 3233.35
```

Example 2.1
An individual wishes to take out a loan, today, of P at a monthly interest rate i. The loan is to be paid back in n monthly installments of size R, beginning one month from now. The problem is to calculate R.

Equating the present value P to the future (discounted) value of the n monthly payments R, we have

$$P = R(1+i)^{-1} + R(1+i)^{-2} + \cdots + R(1+i)^{-n}$$

or

$$P = R\sum_{j=1}^{n}(1+i)^{-j}.$$

Summing this geometric series and simplifying, we obtain

$$P = R\left(\frac{1-(1+i)^{-n}}{i}\right).$$

This is the formula for the present value of an annuity. We can find R, given P, n, and i as

$$R = P\frac{i}{1-(1+i)^{-n}}.$$

In R, we define variables as follows: `principal` to hold the value of P, `intRate` to hold the interest rate, and `n` to hold the number of payments. We will assign the resulting payment value to an object called `payment`.

Of course, we need some numerical values to work with, so we will suppose that the loan amount is \$1500, the interest rate is 1% and the number of payments is 10. The required code is then

```
intRate <- 0.01
n <- 10
principal <- 1500
payment <- principal * intRate / (1 - (1 + intRate)^(-n))
payment

## [1] 158.3731
```

For this particular loan, the monthly payments are \$158.37.

2.1.3 Quitting R

To quit your R session, run

```
q()
```

or choose `Quit RStudio...` from the `File` menu. You will then be asked whether to save an image of the current workspace, or not, or to cancel. The workspace image contains a record of the computations you've done, and may contain some saved results. Hitting the Cancel option allows you to continue your current R session. We rarely save the current workspace image, but occasionally find it convenient to do so.

Note what happens if you omit the parentheses `()` when attempting to quit: