3.12: Summary

Every book that tries to introduce basic programming ideas to novices has to cover roughly the same topics, and in roughly the same order. Mine is no exception, and so in the grand tradition of doing it just the same way everyone else did it, this chapter covered the following topics:

- **Getting started**. We downloaded and installed R and RStudio
- **Basic commands**. We talked a bit about the logic of how R works and in particular how to type commands into the R console (Section@ref(#firstcommand), and in doing so learned how to perform basic calculations using the arithmetic operators `+`, `-`, `*`, `/` and `^`.
- **Introduction to functions.** We saw several different functions, three that are used to perform numeric calculations (`sqrt()`, `abs()`, `round()`, one that applies to text (`nchar()`; Section@ref(#simpletext), and one that works on any variable (`length()`; Section@ref(#veclength). In doing so, we talked a bit about how argument names work, and learned about default values for arguments. (Section@ref(#functionarguments)
- Introduction to variables. We learned the basic idea behind variables, and how to assign values to variables using the assignment operator `<-` (Section@ref(#assign). We also learned how to create vectors using the combine function `c()`. (Section@ref(#vectors)
- Data types. Learned the distinction between numeric, character and logical data; including the basics of how to enter and use each of them. (Sections@ref(#assign) to Sections 3.9
- **Logical operations**.(#logicals) Learned how to use the logical operators `==`, `!=`, `<`, `>`, `<=`, `=>`, `!`, `&` and `|`. And learned how to use logical indexing. (Section 3.10)

We still haven't arrived at anything that resembles a "data set", of course. Maybe the next Chapter will get us a bit closer…

References

R Core Team. 2013. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

13. Source: *Dismal Light* (1968).
14. Although R is updated frequently, it doesn't usually make much of a difference for the sort of work we'll do in this book. In fact, during the writing of the book I upgraded several times, and didn't have to change much except these sections describing the downloading.
15. If you're running an older version of the Mac OS, then you need to follow the link to the "old" page (http://cran.r-project.org/bin/macosx/old/). You should be able to find the installer file that you need at the bottom of the page.
16. Tip for advanced Mac users. You can run R from the terminal if you want to. The command is just "R". It behaves like the normal desktop version, except that help documentation behaves like a "man" page instead of opening in a new window.
17. This is probably no coincidence: the people who design and distribute the core R language itself are focused on technical stuff. And sometimes they almost seem to forget that there's an actual human user at the end. The people who design and distribute RStudio are focused on user interface. They want to make R as usable as possible. The two websites reflect that difference.
18. Seriously. If you're in a position to do so, open up R and start typing. The simple act of typing it rather than "just reading" makes a big difference. It makes the concepts more concrete, and it ties the abstract ideas (programming and statistics) to the actual context in which you need to use them. Statistics is something you *do*, not just something you read about in a textbook.
19. If you're running R from the terminal rather than from RStudio, escape doesn't work: use CTRL-C instead.
20. For advanced users: yes, as you've probably guessed, R is printing out the source code for the function.
21. If you're reading this with R open, a good learning trick is to try typing in a few different variations on what I've done here. If you experiment with your commands, you'll quickly learn what works and what doesn't
22. For advanced users: if you want a table showing the complete order of operator precedence in R, type `?Syntax`. I haven't included it in this book since there are quite a few different operators, and we don't need that much detail. Besides, in practice most people seem to figure it out from seeing examples: until writing this book I never looked at the formal statement of operator precedence for any language I ever coded in, and never ran into any difficulties.
23. If you are using RStudio, and the "environment" panel (formerly known as the "workspace" panel) is visible when you typed the command, then you probably saw something happening there. That's to be expected, and is quite helpful. However, there's

two things to note here (1) I haven't yet explained what that panel does, so for now just ignore it, and (2) this is one of the helpful things RStudio does, not a part of R itself.

24. As we'll discuss later, by doing this we are implicitly using the `print()` function
25. Actually, in keeping with the R tradition of providing you with a billion different screwdrivers (even when you're actually looking for a hammer) these aren't the only options. There's also the `assign()` function, and the `<<-` and `->>` operators. However, we won't be using these at all in this book.
26. A quick reminder: when using operators like `<-` and `->` that span multiple characters, you can't insert spaces in the middle. That is, if you type `- >` or `< -`, R will interpret your command the wrong way. And I will cry.
27. Actually, you can override any of these rules if you want to, and quite easily. All you have to do is add quote marks or backticks around your non-standard variable name. For instance `` `my sales` <- 350 `` would work just fine, but it's almost never a good idea to do this.
28. For very advanced users: there is one exception to this. If you're naming a function, don't use `.` in the name unless you are intending to make use of the S3 object oriented programming system in R. If you don't know what S3 is, then you definitely don't want to be using it! For function naming, there's been a trend among R users to prefer `myFunctionName`.
29. A side note for students with a programming background. Technically speaking, operators *are* functions in R: the addition operator `+` is actually a convenient way of calling the addition function `+()`
30. A note for the mathematically inclined: R does support complex numbers, but unless you explicitly specify that you want them it assumes all calculations must be real valued. By default, the square root of a negative number is treated as undefined: `sqrt(-9)` will produce `NaN` (not a number) as its output. To get complex numbers, you would type `sqrt(-9+0i)` and R would now return `0+3i`. However, since we won't have any need for complex numbers in this book, I won't refer to them again.
31. The two functions discussed previously, `sqrt()` and `abs()`, both only have a single argument, `x`. So I could have typed something like `sqrt(x = 225)` or `abs(x = -13)` earlier. The fact that all these functions use `x` as the name of the argument that corresponds the "main" variable that you're working with is no coincidence. That's a fairly widely used convention. Quite often, the writers of R functions will try to use conventional names like this to make your life easier. Or at least that's the theory. In practice it doesn't always work as well as you'd hope.
32. For advanced users: obviously, this isn't just an RStudio thing. If you're running R in a terminal window, tab autocomplete still works, and does so in exactly the way you'd expect. It's not as visually pretty as the RStudio version, of course, and lacks some of the cooler features that RStudio provides. I don't bother to document that here: my assumption is that if you are running R in the terminal then you're already familiar with using tab autocomplete.
33. Incidentally, that always works: if you've started typing a command and you want to clear it and start again, hit escape.
34. Another method is to start typing some text and then hit the Control key and the up arrow together (on Windows or Linux) or the Command key and the up arrow together (on a Mac). This will bring up a window showing all your recent commands that started with the same text as what you've currently typed. That can come in quite handy sometimes.
35. Notice that I didn't specify any argument names here. The `c()` function is one of those cases where we don't use names. We just type all the numbers, and R just dumps them all in a single variable.
36. I offer up my teenage attempts to be "cool" as evidence that some things just can't be done.
37. Note that this is a very different operator to the assignment operator `=` that I talked about in Section 3.4. A common typo that people make when trying to write logical commands in R (or other languages, since the "`=` versus `==`" distinction is important in most programming languages) is to accidentally type `=` when you really mean `==`. Be especially cautious with this – I've been programming in various languages since I was a teenager, and I *still* screw this up a lot. Hm. I think I see why I wasn't cool as a teenager. And why I'm still not cool.
38. A note for those of you who have taken a computer science class: yes, R does have a function for exclusive-or, namely `xor()`. Also worth noting is the fact that R makes the distinction between element-wise operators `&` and `|` and operators that look only at the first element of the vector, namely `&&` and `||`. To see the distinction, compare the behaviour of a command like `c(FALSE,TRUE) & c(TRUE,TRUE)` to the behaviour of something like `c(FALSE,TRUE) && c(TRUE,TRUE)`. If this doesn't mean anything to you, ignore this footnote entirely. It's not important for the content of this book.
39. Warning! `TRUE` and `FALSE` are reserved keywords in R, so you can trust that they always mean what they say they do. Unfortunately, the shortcut versions `T` and `F` do not have this property. It's even possible to create variables that set up the reverse meanings, by typing commands like `T <- FALSE` and `F <- TRUE`. This is kind of insane, and something that is generally thought to be a design flaw in R. Anyway, the long and short of it is that it's safer to use `TRUE` and `FALSE`.

40. Well, I say that… but in my personal experience it wasn't until I started learning "regular expressions" that my loathing of computers reached its peak.
