4.5: Loading and Saving Data

There are several different types of files that are likely to be relevant to us when doing data analysis. There are three in particular that are especially important from the perspective of this book:

- *Workspace files* are those with a .Rdata file extension. This is the standard kind of file that R uses to store data and variables. They're called "workspace files" because you can use them to save your whole workspace.
- *Comma separated value (CSV) files* are those with a .csv file extension. These are just regular old text files, and they can be opened with almost any software. It's quite typical for people to store data in CSV files, precisely because they're so simple.
- *Script files* are those with a .R file extension. These aren't data files at all; rather, they're used to save a collection of commands that you want R to execute later. They're just text files, but we won't make use of them until Chapter 8.

There are also several other types of file that R makes use of,[52] but they're not really all that central to our interests. There are also several other kinds of data file that you might want to import into R. For instance, you might want to open Microsoft Excel spreadsheets (.xlsx files), or data files that have been saved in the native file formats for other statistics software, such as SPSS, SAS, Minitab, Stata or Systat. Finally, you might have to handle databases. R tries hard to play nicely with other software, so it has tools that let you open and work with any of these and many others. I'll discuss some of these other possibilities elsewhere in this book (Section 7.9), but for now I want to focus primarily on the two kinds of data file that you're most likely to need: .Rdata files and .csv files. In this section I'll talk about how to load a workspace file, how to import data from a CSV file, and how to save your workspace to a workspace file. Throughout this section I'll first describe the (sometimes awkward) R commands that do all the work, and then I'll show you the (much easier) way to do it using Rstudio.

4.5.1 Loading workspace files using R

When I used the `list.files()` command to list the contents of the `/Users/dan/Rbook/data` directory (in Section 4.4.2), the output referred to a file called booksales.Rdata. Let's say I want to load the data from this file into my workspace. The way I do this is with the `load()` function. There are two arguments to this function, but the only one we're interested in is

- `file`. This should be a character string that specifies a path to the file that needs to be loaded. You can use an absolute path or a relative path to do so.

Using the absolute file path, the command would look like this:

```
load( file = "/Users/dan/Rbook/data/booksales.Rdata" )
```

but this is pretty lengthy. Given that the working directory (remember, we changed the directory at the end of Section 4.4.4) is `/Users/dan/Rbook/data`, I could use a relative file path, like so:

```
load( file = "../data/booksales.Rdata" )
```

However, my preference is usually to change the working directory first, and *then* load the file. What that would look like is this:

```
setwd( "../data" )         # move to the data directory
load( "booksales.Rdata" )  # load the data
```

If I were then to type `who()` I'd see that there are several new variables in my workspace now. Throughout this book, whenever you see me loading a file, I will assume that the file is actually stored in the working directory, or that you've changed the working directory so that R is pointing at the directory that contains the file. Obviously, *you* don't need type that command yourself: you can use the Rstudio file panel to do the work.

4.5.2 Loading workspace files using Rstudio

Okay, so how do we open an .Rdata file using the Rstudio file panel? It's terribly simple. First, use the file panel to find the folder that contains the file you want to load. If you look at Figure 4.7, you can see that there are several .Rdata files listed. Let's say I want to load the `booksales.Rdata` file. All I have to do is click on the file name. Rstudio brings up a little dialog box asking me to confirm that I do want to load this file. I click yes. The following command then turns up in the console,

```
load("~/Rbook/data/booksales.Rdata")
```

and the new variables will appear in the workspace (you'll see them in the Environment panel in Rstudio, or if you type `who()`). So easy it barely warrants having its own section.

One quite commonly used data format is the humble "comma separated value" file, also called a CSV file, and usually bearing the file extension .csv. CSV files are just plain old-fashioned text files, and what they store is basically just a table of data. This is illustrated in Figure 4.8, which shows a file called booksales.csv that I've created. As you can see, each row corresponds to a variable, and each row represents the book sales data for one month. The first row doesn't contain actual data though: it has the names of the variables.

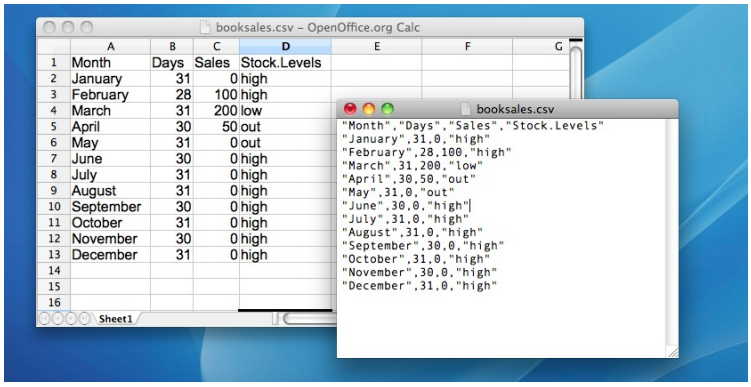

Figure 4.8: The booksales.csv data file. On the left, I've opened the file in using a spreadsheet program (OpenOffice), which shows that the file is basically a table. On the right, the same file is open in a standard text editor (the TextEdit program on a Mac), which shows how the file is formatted. The entries in the table are wrapped in quote marks and separated by commas.

If Rstudio were not available to you, the easiest way to open this file would be to use the `read.csv()` function.[53] This function is pretty flexible, and I'll talk a lot more about it's capabilities in Section 7.9 for more details, but for now there's only two arguments to the function that I'll mention:

- `file`. This should be a character string that specifies a path to the file that needs to be loaded. You can use an absolute path or a relative path to do so.
- `header`. This is a logical value indicating whether or not the first row of the file contains variable names. The default value is `TRUE`.

Therefore, to import the CSV file, the command I need is:

```
books <- read.csv( file = "booksales.csv" )
```

There are two very important points to notice here. Firstly, notice that I *didn't* try to use the `load()` function, because that function is only meant to be used for .Rdata files. If you try to use `load()` on other types of data, you get an error. Secondly, notice that when I imported the CSV file I assigned the result to a variable, which I imaginatively called `books`.[54] file. There's a reason for this. The idea behind an `.Rdata` file is that it stores a whole workspace. So, if you had the ability to look inside the file yourself you'd see that the data file keeps track of all the variables and their names. So when you `load()` the file, R restores all those original names. CSV files are treated differently: as far as R is concerned, the CSV only stores *one* variable, but that variable is big table. So when you import that table into the workspace, R expects *you* to give it a name.] Let's have a look at what we've got:

```
print( books )
```

```
##        Month Days Sales Stock.Levels
## 1    January   31     0         high
## 2   February   28   100         high
## 3      March   31   200          low
## 4      April   30    50          out
## 5        May   31     0          out
## 6       June   30     0         high
## 7       July   31     0         high
## 8     August   31     0         high
## 9  September   30     0         high
## 10   October   31     0         high
## 11  November   30     0         high
## 12  December   31     0         high
```

Clearly, it's worked, but the format of this output is a bit unfamiliar. We haven't seen anything like this before. What you're looking at is a *data frame*, which is a very important kind of variable in R, and one I'll discuss in Section 4.8. For now, let's just be happy that we imported the data and that it looks about right.

4.5.3 Importing data from CSV files using Rstudio

Yet again, it's easier in Rstudio. In the environment panel in Rstudio you should see a button called "Import Dataset". Click on that, and it will give you a couple of options: select the "From Text File…" option, and it will open up a very familiar dialog box asking you to select a file: if you're on a Mac, it'll look like the usual Finder window that you use to choose a file; on Windows it looks like an Explorer window. An example of what it looks like on a Mac is shown in Figure 4.9. I'm assuming that you're familiar with your own computer, so you should have no problem finding the CSV file that you want to import! Find the one you want, then click on the "Open" button. When you do this, you'll see a window that looks like the one in Figure 4.10.

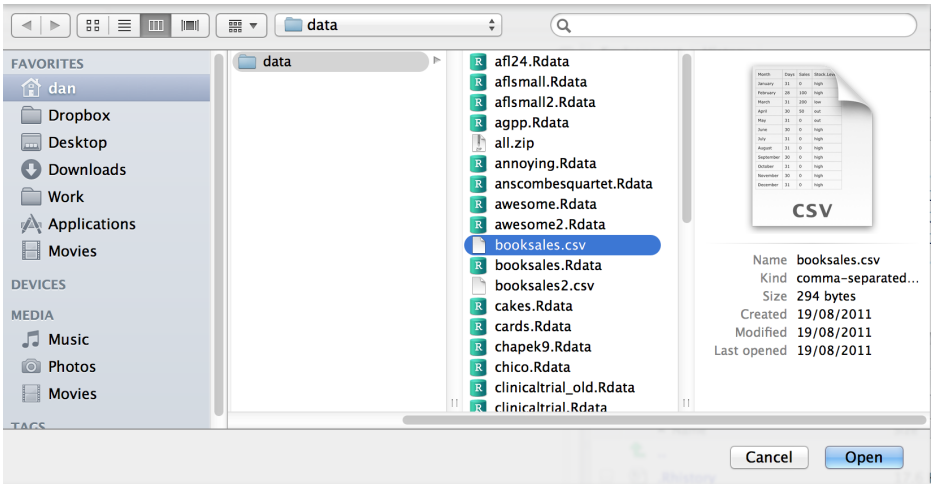

Figure 4.9: A dialog box on a Mac asking you to select the CSV file R should try to import. Mac users will recognise this immediately: it's the usual way in which a Mac asks you to find a file. Windows users won't see this: they'll see the usual explorer window that Windows always gives you when it wants you to select a file.

The import data set window is relatively straightforward to understand.

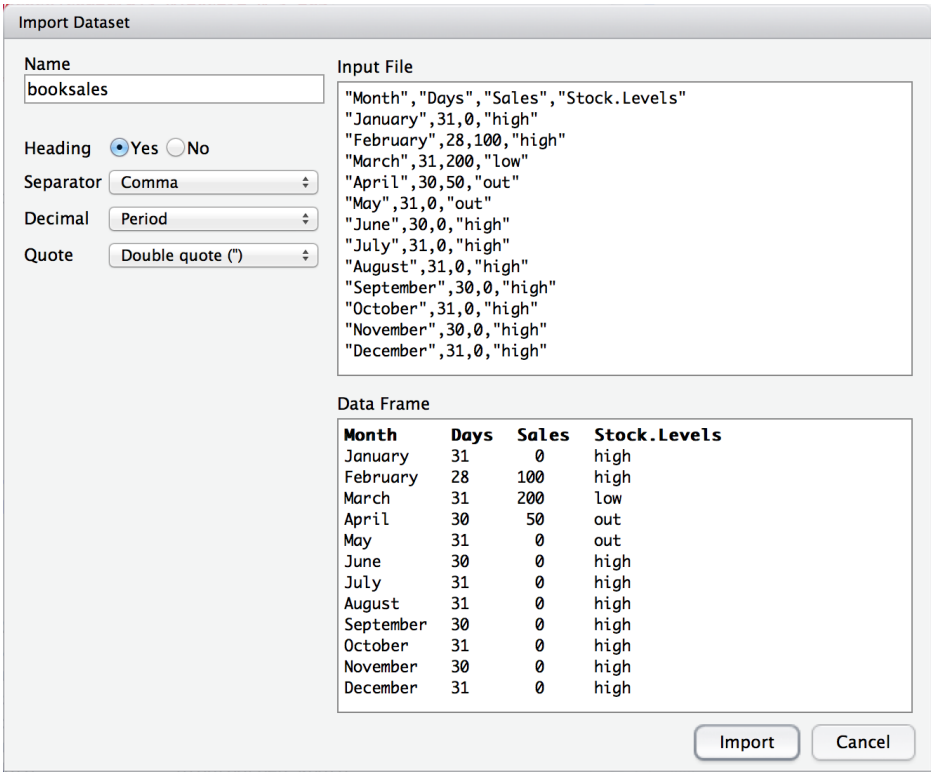

Figure 4.10: The Rstudio window for importing a CSV file into R

In the top left corner, you need to type the name of the variable you R to create. By default, that will be the same as the file name: our file is called `booksales.csv`, so Rstudio suggests the name `booksales`. If you're happy with that, leave it alone. If not, type something else. Immediately below this are a few things that you can tweak to make sure that the data gets imported correctly:

- Heading. Does the first row of the file contain raw data, or does it contain headings for each variable? The `booksales.csv` file has a header at the top, so I selected "yes".
- Separator. What character is used to separate different entries? In most CSV files this will be a comma (it is "comma separated" after all). But you can change this if your file is different.
- Decimal. What character is used to specify the decimal point? In English speaking countries, this is almost always a period (i.e., `.`). That's not universally true: many European countries use a comma. So you can change that if you need to.
- Quote. What character is used to denote a block of text? That's usually going to be a double quote mark. It is for the `booksales.csv` file, so that's what I selected.

The nice thing about the Rstudio window is that it shows you the raw data file at the top of the window, and it shows you a preview of the data at the bottom. If the data at the bottom doesn't look right, try changing some of the settings on the left hand side. Once you're happy, click "Import". When you do, two commands appear in the R console:

```
booksales <- read.csv("~/Rbook/data/booksales.csv")
View(booksales)
```

The first of these commands is the one that loads the data. The second one will display a pretty table showing the data in Rstudio.

4.5.4 Saving a workspace file using `save`

Not surprisingly, saving data is very similar to loading data. Although Rstudio provides a simple way to save files (see below), it's worth understanding the actual commands involved. There are two commands you can use to do this, `save()` and `save.image()`. If you're happy to save *all* of the variables in your workspace into the data file, then you should use `save.image()`. And if you're happy for R to save the file into the current working directory, all you have to do is this:

```
save.image( file = "myfile.Rdata" )
```

Since `file` is the first argument, you can shorten this to `save.image("myfile.Rdata")` ; and if you want to save to a different directory, then (as always) you need to be more explicit about specifying the path to the file, just as we discussed in Section 4.4. Suppose, however, I have several variables in my workspace, and I only want to save some of them. For instance, I might have this as my workspace:

```
who()
##   -- Name --    -- Class --    -- Size --
##   data          data.frame     3 x 2
##   handy         character      1
##   junk          numeric        1
```

I want to save `data` and `handy` , but not `junk` . But I don't want to delete `junk` right now, because I want to use it for something else later on. This is where the `save()` function is useful, since it lets me indicate exactly which variables I want to save. Here is one way I can use the `save` function to solve my problem:

```
save(data, handy, file = "myfile.Rdata")
```

Importantly, you *must* specify the name of the `file` argument. The reason is that if you don't do so, R will think that `"myfile.Rdata"` is actually a *variable* that you want to save, and you'll get an error message. Finally, I should mention a second way to specify which variables the `save()` function should save, which is to use the `list` argument. You do so like this:

```
save.me <- c("data", "handy")   # the variables to be saved
save( file = "booksales2.Rdata", list = save.me )   # the command to save them
```

4.5.5 Saving a workspace file using Rstudio

Rstudio allows you to save the workspace pretty easily. In the environment panel (Figures 4.5 and 4.6) you can see the "save" button. There's no text, but it's the same icon that gets used on every computer everywhere: it's the one that looks like a floppy disk. You know, those things that haven't been used in about 20 years. Alternatively, go to the "Session" menu and click on the "Save Workspace As…" option.[55] This will bring up the standard "save" dialog box for your operating system (e.g., on a Mac it'll look a little bit like the loading dialog box in Figure 4.9). Type in the name of the file that you want to save it to, and all the variables in your workspace will be saved to disk. You'll see an R command like this one

```
save.image("~/Desktop/Untitled.RData")
```

Pretty straightforward, really.

4.5.6 Other things you might want to save

Until now, we've talked mostly about loading and saving *data*. Other things you might want to save include:

- *The output*. Sometimes you might also want to keep a copy of all your interactions with R, including everything that you typed in and everything that R did in response. There are some functions that you can use to get R to write its output to a file rather than to print onscreen (e.g., `sink()`), but to be honest, if you do want to save the R output, the easiest thing to do is to use the mouse to select the relevant text in the R console, go to the "Edit" menu in Rstudio and select "Copy". The output has now been copied to the clipboard. Now open up your favourite text editor or word processing software, and paste it. And you're done. However, this will only save the contents of the console, not the plots you've drawn (assuming you've drawn some). We'll talk about saving images later on.
- *A script*. While it is possible – and sometimes handy – to save the R output as a method for keeping a copy of your statistical analyses, another option that people use a lot (especially when you move beyond simple "toy" analyses) is to write *scripts*. A

script is a text file in which you write out all the commands that you want R to run. You can write your script using whatever software you like. In real world data analysis writing scripts is a key skill – and as you become familiar with R you'll probably find that most of what you do involves scripting rather than typing commands at the R prompt. However, you won't need to do much scripting initially, so we'll leave that until Chapter 8.
