4.4: Navigating the File System

In this section I talk a little about how R interacts with the file system on your computer. It's not a terribly interesting topic, but it's useful. As background to this discussion, I'll talk a bit about how file system locations work in Section 4.4.1. Once upon a time *everyone* who used computers could safely be assumed to understand how the file system worked, because it was impossible to successfully use a computer if you didn't! However, modern operating systems are much more "user friendly", and as a consequence of this they go to great lengths to hide the file system from users. So these days it's not at all uncommon for people to have used computers most of their life and not be familiar with the way that computers organise files. If you already know this stuff, skip straight to Section 4.4.2. Otherwise, read on. I'll try to give a brief introduction that will be useful for those of you who have never been forced to learn how to navigate around a computer using a DOS or UNIX shell.

4.4.1 file system itself

In this section I describe the basic idea behind file locations and file paths. Regardless of whether you're using Window, Mac OS or Linux, every file on the computer is assigned a (fairly) human readable address, and every address has the same basic structure: it describes a *path* that starts from a *root* location , through as series of *folders* (or if you're an old-school computer user, *directories*), and finally ends up at the file.

On a Windows computer the root is the physical drive[50] on which the file is stored, and for most home computers the name of the hard drive that stores all your files is C: and therefore most file names on Windows begin with C:. After that comes the folders, and on Windows the folder names are separated by a `\` symbol. So, the complete path to this book on my Windows computer might be something like this:

```
C:\Users\danRbook\LSR.pdf
```

and what that *means* is that the book is called LSR.pdf, and it's in a folder called `book` which itself is in a folder called dan which itself is … well, you get the idea. On Linux, Unix and Mac OS systems, the addresses look a little different, but they're more or less identical in spirit. Instead of using the backslash, folders are separated using a forward slash, and unlike Windows, they don't treat the physical drive as being the root of the file system. So, the path to this book on my Mac might be something like this:

```
/Users/dan/Rbook/LSR.pdf
```

So that's what we mean by the "path" to a file. The next concept to grasp is the idea of a ***working directory*** and how to change it. For those of you who have used command line interfaces previously, this should be obvious already. But if not, here's what I mean. The working directory is just "whatever folder I'm currently looking at". Suppose that I'm currently looking for files in Explorer (if you're using Windows) or using Finder (on a Mac). The folder I currently have open is my user directory (i.e., `C:\Users\dan` or `/Users/dan`). That's my current working directory.

The fact that we can imagine that the program is "in" a particular directory means that we can talk about moving *from* our current location *to* a new one. What that means is that we might want to specify a new location in relation to our current location. To do so, we need to introduce two new conventions. Regardless of what operating system you're using, we use . to refer to the current working directory, and .. to refer to the directory above it. This allows us to specify a path to a new location in relation to our current location, as the following examples illustrate. Let's assume that I'm using my Windows computer, and my working directory is `C:\Users\danRbook`). The table below shows several addresses in relation to my current one:

Table 4.1: Basic arithmetic operations in R. These five operators are used very frequently throughout the text, so it's important to be familiar with them at the outset.

absolute path (i.e., from root)	**relative path (i.e. from C:)**
C:\Users\dan	..
C:\Users	..\.. \
C:\Users\danRbook\source	.\source
C:\Users\dan\nerdstuff	..\nerdstuff

There's one last thing I want to call attention to: the `~` directory. I normally wouldn't bother, but R makes reference to this concept sometimes. It's quite common on computers that have multiple users to define `~` to be the user's home directory. On my Mac, for instance, the home directory `~` for the "dan" user is `\Users\dan\`. And so, not surprisingly, it is possible to define other directories in terms of their relationship to the home directory. For example, an alternative way to describe the location of the `LSR.pdf` file on my Mac would be

```
~Rbook\LSR.pdf
```

That's about all you really need to know about file paths. And since this section already feels too long, it's time to look at how to navigate the file system in R.

4.4.2 Navigating the file system using the R console

In this section I'll talk about how to navigate this file system from within R itself. It's not particularly user friendly, and so you'll probably be happy to know that Rstudio provides you with an easier method, and I will describe it in Section 4.4.4. So in practice, you won't *really* need to use the commands that I babble on about in this section, but I do think it helps to see them in operation at least once before forgetting about them forever.

Okay, let's get started. When you want to load or save a file in R it's important to know what the working directory is. You can find out by using the `getwd()` command. For the moment, let's assume that I'm using Mac OS or Linux, since there's some subtleties to Windows. Here's what happens:

```
getwd()
## [1] "/Users/dan"
```

We can change the working directory quite easily using `setwd()`. The `setwd()` function has only the one argument, `dir`, is a character string specifying a path to a directory, or a path relative to the working directory. Since I'm currently located at `/Users/dan`, the following two are equivalent:

```
setwd("/Users/dan/Rbook/data")
setwd("./Rbook/data")
```

Now that we're here, we can type `list.files()` command to get a listing of all the files in that directory. Since this is the directory in which I store all of the data files that we'll use in this book, here's what we get as the result:

```
list.files()
## [1] "afl24.Rdata"             "aflsmall.Rdata"          "aflsmall2.Rdata"
## [4] "agpp.Rdata"              "all.zip"                 "annoying.Rdata"
## [7] "anscombesquartet.Rdata"  "awesome.Rdata"           "awesome2.Rdata"
## [10] "booksales.csv"           "booksales.Rdata"         "booksales2.csv"
## [13] "cakes.Rdata"             "cards.Rdata"             "chapek9.Rdata"
## [16] "chico.Rdata"             "clinicaltrial_old.Rdata" "clinicaltrial.Rdata"
## [19] "coffee.Rdata"            "drugs.wmc.rt.Rdata"      "dwr_all.Rdata"
## [22] "effort.Rdata"            "happy.Rdata"             "harpo.Rdata"
## [25] "harpo2.Rdata"            "likert.Rdata"            "nightgarden.Rdata"
## [28] "nightgarden2.Rdata"      "parenthood.Rdata"        "parenthood2.Rdata"
## [31] "randomness.Rdata"        "repeated.Rdata"          "rtfm.Rdata"
## [34] "salem.Rdata"             "zeppo.Rdata"
```

Not terribly exciting, I'll admit, but it's useful to know about. In any case, there's only one more thing I want to make a note of, which is that R also makes use of the home directory. You can find out what it is by using the `path.expand()` function, like this:

```
path.expand("~")
## [1] "/Users/dan"
```

You can change the user directory if you want, but we're not going to make use of it very much so there's no reason to. The only reason I'm even bothering to mention it at all is that when you use Rstudio to open a file, you'll see output on screen that defines the path to the file relative to the #~# directory. I'd prefer you not to be confused when you see it.[51]

4.4.3 the Windows paths use the wrong slash?

Let's suppose I'm on Windows. As before, I can find out what my current working directory is like this:

```
getwd()
## [1] "C:/Users/dan/
```

This seems about right, but you might be wondering why R is displaying a Windows path using the wrong type of slash. The answer is slightly complicated, and has to do with the fact that R treats the `\` character as "special" (see Section 7.8.7). If you're deeply wedded to the idea of specifying a path using the Windows style slashes, then what you need to do is to type `/` whenever you mean `\`. In other words, if you want to specify the working directory on a Windows computer, you need to use one of the following commands:

```
setwd( "C:/Users/dan" )
setwd( "C:\\Users\\dan" )
```

It's kind of annoying to have to do it this way, but as you'll see later on in Section 7.8.7 it's a necessary evil. Fortunately, as we'll see in the next section, Rstudio provides a much simpler way of changing directories…

4.4.4 Navigating the file system using the Rstudio file panel

Although I think it's important to understand how all this command line stuff works, in many (maybe even most) situations there's an easier way. For our purposes, the easiest way to navigate the file system is to make use of Rstudio's built in tools. The "file" panel – the lower right hand area in Figure 4.7 – is actually a pretty decent file browser. Not only can you just point and click on the names to move around the file system, you can also use it to set the working directory, and even load files.

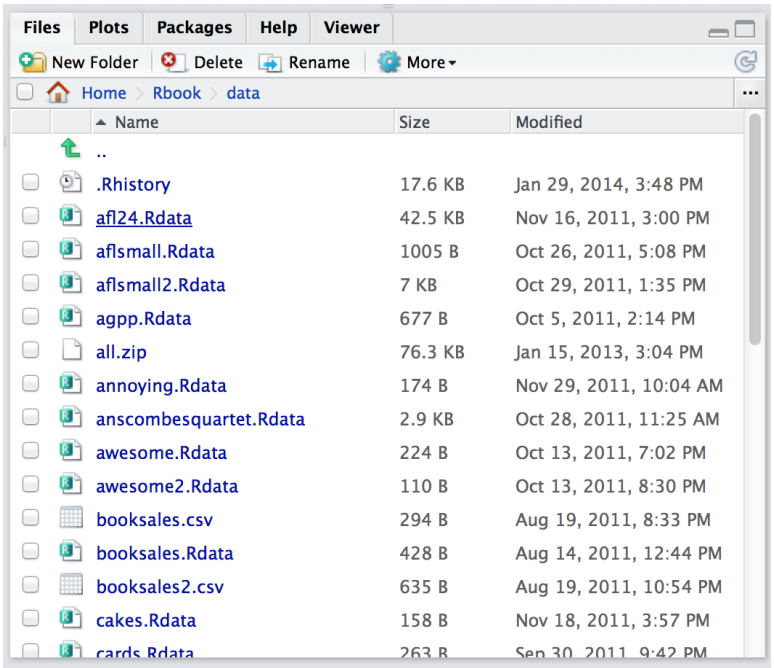

Figure 4.7: The "file panel" is the area shown in the lower right hand corner. It provides a very easy way to browse and navigate your computer using R. See main text for details.

Here's what you need to do to change the working directory using the file panel. Let's say I'm looking at the actual screen shown in Figure 4.7. At the top of the file panel you see some text that says "Home > Rbook > data". What that means is that it's *displaying* the files that are stored in the

```
/Users/dan/Rbook/data
```

directory on my computer. It does *not* mean that this is the R working directory. If you want to change the R working directory, using the file panel, you need to click on the button that reads "More". This will bring up a little menu, and one of the options will be "Set as Working Directory". If you select that option, then R really will change the working directory. You can tell that it has done so because this command appears in the console:

```
setwd("~/Rbook/data")
```

In other words, Rstudio sends a command to the R console, exactly as if you'd typed it yourself. The file panel can be used to do other things too. If you want to move "up" to the parent folder (e.g., from `/Users/dan/Rbook/data` to `/Users/dan/Rbook` click on the ".." link in the file panel. To move to a subfolder, click on the name of the folder that you want to open. You can open some types of file by clicking on them. You can delete files from your computer using the "delete" button, rename them with the "rename" button, and so on.

As you can tell, the file panel is a very handy little tool for navigating the file system. But it can do more than just navigate. As we'll see later, it can be used to open files. And if you look at the buttons and menu options that it presents, you can even use it to rename, delete, copy or move files, and create new folders. However, since most of that functionality isn't critical to the basic goals of this book, I'll let you discover those on your own.
