Pre-class Video Scripts

DECS 922

Professor Robert McDonald

Winter 2019

Contents

1 Introduction

These scripts are notes I used when creating the screencasts, not actual transcripts. But you may find them helpful in following along.

2 Introduction to RStudio

The purpose of this video is to introduce you to RStudio, which is a tool we will use throughout the course. RStudio provides an interface to R and also integrates other software that makes R more useful. You should watch this video with RStudio running. It would be good for you to periodically stop and perform the same actions that you see me perform.

2.1 Newly-opened RStudio

When you first open RStudio, it will look something like this. There are 3 windows. There is a single window on the left. There are two windows on the right, each of which has several tabs at the top. The window on the right at the top has the "environment" tab selected, and the window on the bottom has the "packages" tab selected. You can make the same selections now.

The window on the left is called the "console". This provides a space where you can type R commands and they will be executed. Type Ctrl-L (that is hold down the ctrl key and type the letter l) now. This clears the window.

If you type a command that creates printed output, you will see the output. Let's look at some examples:

```
5*3
sqrt(81)
x <- 3
```

Notice that in the last example I created a variable, `x`, and assigned the number "3" to it. I can use `x` as if it were a number:

```
5*x
```

Notice also that every time RStudio prints a result, there is a "1" in square brackets at the start of the line. Everything in R is a vector, which we'll discuss later. The 1 in brackets tells you that the value being printed is the first element of the vector. You can just ignore it for the time being and we'll discuss this later.

R has a full complement of mathematical functions, such as the square root function. We won't try to list them.

2.2 Environment and History

On the top right, let's look at the "environment" panel. If it is not showing, select it now by clicking on its tab. Notice that there is an entry that has `x` on the left and `5` on the right. Now type `x=80`. You will see the line for `x` update to show the new value.

There is also a history tab. If you click on it, you will see the commands you have typed in the past. If you click on an entry and hit `<enter>`, the command will appear in the console on the left. If you hit `<enter>` again, the command will execute. The storing of commands in the history pane is automatic.

You can also execute blocks of commands. Click on "`5*3`". Hold down the shift key and click on "`5*x`" Now hit the `<enter>` key, or else click the little icon on the top that says "To Console". You will see the commands appear in the console. Press `<enter>` and the block will execute.

2.3 Help etc.

On the bottom right you have a window that will show you files in your current working directory, plots, which packages you have installed (and which versions).

There is an options dialog that allows you to customize much of what you see. Go to Tools|Global Options|Pane Layout and have a look.

2.4 Keyboard shortcuts

RStudio has a number of keyboard shortcuts for various actions. If you click on "Help" followed by "Keyboard Shortcuts Help", you will see a window that shows you the keyboard shortcuts for your operating system.

There is a link in this window that says "See all shortcuts". This will take you to a webpage that shows you shortcuts for Windows and Linux, which use the same shortcuts, side-by-side with those for OS X.

This concludes our introduction to RStudio.

3 Introduction to the RStudio editor

In the previous video we looked at RStudio and saw that there are 3 windows visible at startup. There is a fourth window, the editor window, that appears when you are editing an R script or a document.

In this video I am going to show you what a document looks like in RStudio, and I will show you how it is constructed. We will be using such documents throughout the course. This video will demonstrate one of the most exciting aspects of R and RStudio.

3.1 The Editor window

RStudio has 4 windows, but the editor window is visible only when you have a document to edit. Let's examine an Rmarkdown document, which is the kind of document we will be using throughout the course.

Look at the menu items along the top of the RStudio window. One says "File". Select File, then New File, then RMarkdown. Give your document a title, then click "OK". You will see a new document open in the editor window at the top left. This is the default RMarkdown document, created by RStudio. You will see this every time you create a new Rmarkdown document. It is a shell, designed to provide some structure so it is easy to replace the pre-populated content with your own content. You can delete everything if you wish, but we will leave everything in place for the moment. Let's talk about what we're looking at.

3.2 RMarkdown documents

There are three basic components to an RMarkdown document.

1. YAML header. There is information at the top, called the "YAML Header". This describes the document – it's title, author, etc. This is sometimes referred to as document metadata, because it is data about the document, not related to the content of the document.
2. R Code. There is R code that is contained within regions demarcated by sets of three consecutive backticks starting in column 1. These code

sections are called "chunks", and are executed when you click the "knit" button.
3. Text. There is standard text throughout, along with text formatted using Markdown.

3.2.1 Yaml header

At the top of the document you will see three dashes, followed by lines which describe the title, author, date, and output format. This is the "YAML header". (YAMl is spelled Y-A-M-L: Yet Another Markup Language). The header contains information about the document. You can include a lot more information than this in the YAML header. But for now this is sufficient.

You may think the YAML header looks primitive. YAML is actually a recognized data format for a "document oriented database", which is a type of NoSQL database. We may have time later in the course to discuss such databases.

3.2.2 R Code Chunks

If you scroll down from the YAML header you'll see three backticks, followed by some text in braces, followed by some code, followed by three more backticks. This is an R chunk. What you see there may look mysterious at this point. No worries. We'll understand it later in the course. What you should understand now is that a chunk can contain *any* valid R code, and it can be many lines long.

3.2.3 Markdown text

The content that is neither YAML nor R chunks is text, in a format called "markdown". Below the R chunk, you see two hash signs followed by "R Markdown". When we execute this document by "knitting" it, the two hash signs will mean "make this a subsection heading".

If you click on "Help|Markdown Quick Reference", on the main menu at the top, you will see in the help window a brief introduction to markdown. There is a more elaborate set of help pages on the RStudio web site, and there is

also a one page "cheatsheet" which is quite good but better as a reference once you understand the basics.

3.2.4 Knitting

We have seen that your RMarkdown document has three components: A YAML header, R code chunks, and markdown text. You can create a document by clicking the "knit" button, as you did when testing your setup. Doing this will create an HTML document that opens in an HTML browser window. It is instructive to look at the HTML document and compare it to the contents of the RMarkdown document. You can do that at your leisure.

How does RStudio convert the Rmarkdown document into html? There are several steps. First, Rstudio calls R to run all the R code in the document. It converts the code and output into markdown. Next, RStudio calls pandoc, which can converts markdown into many different formats. In this case, the markdown is converted into HTML.

You are not restricted to HTML output. Pandoc has a rich variety of output formats. There is a little down arrow next to the Knit button. You can create a pdf file by selecting "Knit to PDF". (You need LaTeX installed for this.) You can create a Word document by selecting "Knit to Word". You should give both a try. This is all quite magical the first time you see it, but it is a powerful and efficient way to record the steps and convey the results of your analysis.

3.2.5 Adding to the document

As a final exercise, we will add one chunk to this document. Go to the end. Click "insert", and then R. You will see a blank chunk. You can also create a blank chunk with a keyboard shortcut: Alt+Ctrl+I or on the Mac, Command + Option + I. Create a blank chunk that looks like the following:

```
hist(mtcars$cyl)
```

When you knit the results, you will see a histogram showing the distribution in the Motor Trend cars data of the number of cylinders.

There are many different "chunk options" which add captions, suppress the printing of R code, reuse code, control evaluation of the code, and so forth. For a good time, can google "R chunk options" and explore the chunk option documentation.

3.3 Conclusion

Rmarkdown is just the beginning. Many authors have written books using an enhanced version of RMarkdown called bookdown. There is a blogdown package for creating static web sites.

I think you'll find that R and RStudio have extraordinary publishing capabilities.

4 Introduction to vectors

R has three data structures that you need to understand: vectors, dataframes, and lists. There are other data structures, but these three are the most important for data analysis. In this video I will discuss vectors.

You should already be familiar with arrays in Excel. A vector is like a one-dimensional array in Excel, a collection of data items all in one column or all in one row in adjacent cells. A vector in R is analogous, and also has the chracteristic that every element has the same type. That is, all elements are numeric, all are character, all are logical, etc. The types are not mixed. When your data storage needs become more complicated than this, you turn to dataframes and lists.

We'll begin by creating two numeric vectors. Go to the console in RStudio and enter this:

```
x <- c(1, 10, 100.01, 1000)
y <- 8:11
```

You have just created two R objects, each of which is a vector containing 4 numbers. The first, `x`, stores the numbers 1, 10, 100.01, and 1000. The expression `c(1, 10, 100.01, 1000)` means concatenate the numbers 1, 10, 100.10, and 1000. (The dictionary definition of concatenate is to link things together in a chain or series.) By creating a vector, you are linking these numbers together in one place, namely the vector "x".

The second, vector, `y`, stores the numbers 8, 9, 10, and 11. The expression `8 colon 11` is shorthand for the sequence created by starting at 8, ending at 11, incrementing by one in the middle.

`x` and `y` are examples of *numerical vectors*. Each is an R object storing multiple items of the same type. And in this case each is storing numbers.

I want to illustrate, with a few examples, how you can manipulate vectors. These examples should help you get started. You can easily play around with such examples on your own.

Here are some important things to know about vectors:

- Most of the time you can use a vector in a function or calculation and

you will get a vector result

```
x
x^2
sqrt(x)
```

- You can extract specific elements of a vector by using a different vector (e.g. `x[c(2, 4)]` instructs R to pick out the second and fourth element of the vector `x`.

```
x[2]
x[2:4]
x[c(2, 4)]
```

- Notice that we used square brackets to denote vector indexes. Single and double square bracket, but we used parentheses when
- You can delete elements by prefacing the vector with a minus sign

```
x[-3]
x[-c(2, 4)]
```

- If you add, multiply, or divide two vectors, they are summed, multiplied, or divided element-by-element

```
x + y
x*y
x/y
x[c(1, 3)]/y[c(1, 3)]
```

- You can do arithmetic with vectors even if they are different lengths. This is actually a controversial feature of R, which is called the "recycling rule": the shorter vector is "recycled" until the calculation is finished.

```
x/y[c(1, 3)]  ## !! what just happened here????
```

- You can concatenate vectors; everything is linked together.

```
w <- c(x, y, 42, 9)
w
```

There will be more about vectors as we proceed through the course. But these are so fundamental that I wanted to provide an introduction at the outset.