Learning R Markdown

Sean Raleigh

Introduction

This document will teach you about using R Markdown to create quality documents that incorporate text and R code seamlessly. By the end of this tutorial, you will have created your first R Markdown document.

Create a new project

Go to the upper-right corner of the screen. Click the arrow next to the project name to access the project drop-down menu and select "New Project". Generally, you will then select "New Directory". (The only exception would be if you have already created a folder somewhere in which you want your R Markdown file to reside. In that case, click "Existing Directory".) Next, click "Empty Project", and then give your project a name where is asks for "Directory name". When you're done, click "Create Project".

RStudio will switch you to your new working directory. There will already be one file in that directory with a `.Rproj` extension. (The name of the file is whatever you named your new project.) **You should never touch that file!** The `.Rproj` file is an auxiliary file that keeps track of background stuff relating to your project. You will have no need to interact with this file. Just leave it sitting there in your project folder.

Open a new R Markdown file

There are two ways to open a new R Markdown file:

- go to the "File" menu, select "New File", and then "R Markdown", or
- click the button right below the File menu that looks like a plus sign over a white rectangle. Then select "R Markdown" from the drop-down list.

The first time you do this, RStudio may tell you that certain packages are required in order to process an R Markdown document. You should allow RStudio to install the suggested packages.

You will also see a dialog box that allows you to give your document a title and an author. Please fill out both. Your page needs a title, of course, but it especially needs an author so that I can identify your file when I'm grading it. One more option is to specify the format for your output. The default option is HTML and there are options for PDF and Word documents as well. **I strongly recommend that you leave HTML selected for now.** We'll come back to this issue later.

Once you have filled in the options, click "OK".

Save your file!

You should now be looking at a sample file that shows some of the things you can do with R Markdown. The first thing we **always** do is save our file. This is in the "File" menu, but it's easier to just hit Ctrl-S (or cmd-S on a Mac). Give the file a specific name. It will probably be the same or similar to the title you chose when setting up the new project. The only thing to remember is that file names should not have any spaces in them. (In fact, you should avoid other kinds of special character as well, like periods, commas, number signs, etc. Stick to letters and numerals and you should be just fine.) If you want a multiword file name, I recommend using underscores like this: `this_filename_has_spaces_in_it`.

Knitting a document

Going from the raw text of an R Markdown document to formatted output is called "knitting". There is a button in the toolbar right about the text that says "Knit HTML". Go ahead and push it. See what happens.

Once the pretty output is generated, take a few moments to look back and forth between it and the original R Markdown text file. You can see some tricks that we won't need much (embedding web links, for example) and some tricks that we will use everyday (like R code chunks).

YAML header

The section at the top of your R Markdown file is called the YAML header. (Google it if you really care why.) It probably looks something like this:

```
---
title:"This is my file"
author: "Sean Raleigh"
date: "August 24, 2015"
output: html_document
---
```

This header is pre-populated with the information you typed when you first set up your file. You can change anything here that you need.

Make it your own

Select everything below the YAML header and delete it (everything below the three hyphens). Now you have a nearly blank slate. Here are a few things to get you started.

Section headers are created by using `#`. Try typing this into your R Markdown document and then knitting it.

```
# Big heading

## Still pretty big

### Not quite as big
```

Since `#` makes a header as big as the title (too big), I prefer to use `##` for section headers and `###` for subsections.

You can make text *italic* or **bold** by using asterisks. Try this in your R Markdown document. (Don't forget to knit to see the results.)

```
This sentence has *italics* and **bold**.
```

There are many more formatting tricks available. For a good resource on all R Markdown stuff, click on this link for a "cheat sheet".

R code chunks

The most powerful feature of R Markdown is the ability to do data analysis right inside the document. This is accomplished by including R code chunks. An R code chunk doesn't just show you the R code in your output file. It also runs that code and generates output that appears right below the code chunk when you knit the file.

An R code chunk starts with three "backticks" followed by the letter `r` enclosed in braces, and it ends with three more backticks. (The backtick is usually in the upper-left corner of your keyboard, next to the number 1 and sharing a key with the tilde ~.) Observe:

````
```{r}
[r code goes in here]
```
````

Let's try doing something with an R code chunk.

The first code chunk that appears in your document should usually load any packages you need. Type the following in your own R Markdown document and then knit:

````
```{r}
library(mosaic)
```
````

In your output document, you'll see the `library(mosaic)` command, followed by all sorts of stuff that probably looks something like this:

```
## Loading required package: car
## Loading required package: dplyr
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
##
## Loading required package: lattice
## Loading required package: ggplot2
## Loading required package: mosaicData
##
## Attaching package: 'mosaic'
##
## The following objects are masked from 'package:dplyr':
##
##     count, do, tally
##
## The following object is masked from 'package:car':
##
##     logit
##
```

```
## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cov, D, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var
##
## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

(If this doesn't work, it's probably because you haven't installed the mosaic package. Be sure to do that using the command `install.packages("mosaic")` from the console.)

As advertised, in your output you should see not only the R code, but also the result of running the R code. In this case, the output is very boring because it just consists of messages that are generated when trying to load the package. Try changing your code chunk to look like this instead:

````
```{r, message=FALSE}
library(mosaic)
```
````

That should get rid of the annoying messages that are generated. Keep in mind that this is one of the few times we will actively try to hide the output. In general, the whole point is to see both the R code and its output simultaneously.

Okay, let's do something interesting now. Let's look at the `mtcars` data set. Type the following code chunk into your R Markdown document and then knit:

````
```{r}
head(mtcars)
tail(mtcars)
str(mtcars)
summary(mtcars)
```
````

The commands are shown along with the output they generate.

What about plotting? Try this:

````
```{r}
qplot(mtcars$disp, mtcars$mpg)
```
````

You can change the size of the figure using "chunk options". (We've already seen an example of a chunk option above, when we added `message=FALSE`.) The relevant options for graphical output are `fig.width` and `fig.height`. Their default values are 7 (inches), but this can be changed. Try this and compare to the version above:

````
```{r, fig.width = 10, fig.height = 5}
qplot(mtcars$disp, mtcars$mpg)
```
````

Inline R commands

You don't need a standalone R code chunk to do computations. One neat feature is the ability to use R to calculate things right in the middle of your text.

Here's an example. Suppose I wanted to compute the mean mpg for the cars in the `mtcars` dataset. I could do this, of course:

````
```{r}
mean(mtcars$mpg)
```
````

But we can also do this inline. Type the following and knit to see the results:

```
The mean mpg for cars in the `mtcars` dataset is `r mean(mtcars$mpg)`.
```

Notice that in addition to the inline R command that calculated the mean, I also enclosed `mtcars` in backticks to make it stand out as code in the output.

Generating a PDF

Up to this point, knitting has produced an HTML document. HTML is the basic language of websites. The file that was generated has a `.html` extension and could be opened up in any web broswer. That's all fine and good, and it's quick and easy for authoring a page. However, sometimes you will want a PDF docucment for the final, polished document. PDF files tend to be easier for other people to save on their machines and then open and read them.

To get a PDF document, go to the arrow next to the "Knit HTML" button. This will drop down a menu and you can select "Knit PDF" instead. Try that now.

You will see that the basic idea is the same, but some of the formatting looks a little different from the HTML file you have been generating up until now.

My recommendation is to use HTML as you're working due to its speed and ease. Then, when you're ready to publish your final document, switch to PDF.

Conclusion

That's it! Now you know how to write an R Markdown file and use it to generate pretty output files. If you look in your project folder, you should see four files. One is the `.Rproj` file that you were instructed never to touch. The one with extension `.rmd` is your R Markdown file. It is really nothing more than a text file; you could open it up in the most basic notepad program on your computer with no trouble. Then you also have a `.html` file and a `.pdf` file that are the pretty output files generated when you knit to HTML and knit to PDF respectively.