Writing with RMarkdown

Francis L. Yuen

2019-12-11

Learning goals:

1. Why we should use RMarkdown
2. How to combine R with your manuscript
3. How to generate different outputs
4. How to automatically generate APA formatted papers

Before we start, we need to install a few packages:

```
install.packages("devtools")
install.packages("knitr")
devtools::install_github("crsh/papaja")
```

When you install these, it may ask if you want to update some of your current packages. I recommend that you **do not** update anything as it may create problems that I cannot fix for you…

You will also need to install **TeX**. Here are the versions I recommend:

For Windows:
MikTeX
For Macs:
MacTeX

Why use RMarkdown?

What is RMarkdown anyways?

RMarkdown is a special file format that lets you generate cool documents. It is much less user friendly than something like Word, but it is also **much more powerful** once you get over the learning hump.

Pros	Cons
- Automatically does APA formatting for you	- Very spooky at first
- Automatically cites and generates a reference page	- Can run into problems you don't know how to solve
- The same file can output PDF and Word files	- You might be too cool and intimidate your peers
- Reproducible!	

The best way to learn RMarkdown, just like R, is to actually prac-

tice. Let's create a new markdown file by going to File -> New File -> RMarkdown. Enter any title you want and put your name as the author.

When you press 'Knit' for the first time, it will ask you where you want to save your RMarkdown file. Remember the reproducible workflow from before? Make a folder for where you want to store this markdown file AND all your data.

The Basics

There are many parts to an RMarkdown file. The first one is the header, which is the part at the top sandwiched between the `---`. There are a ton of fun things you can put here. For now, press the 'Knit' button and see what happens.

Now, change where it says `html_document` to `word_document` and knit again. Pretty cool, right?

Change it back to `html_document` for now. Notice how there are some #'s infront of certain text? What did those do when you knitted your document?

One single # changes the text that follows it into a first-level heading. Two ## changes it to a second-level heading. Utilizing these will allow you create organized and dynamic documents and reports. For example, make some random headings, then adjust your header section so it looks like this:

The indentation has to be correct here. Each level is separated by one `tab` indent, or two spaces

```
output:
  html_document:
    toc: true
```

Pretty neat, right? Now change it to this:

```
output:
  html_document:
    toc: true
    toc_float: true
```

Note: Different file formats will have different header options.

There are tons of other fancy things you can do. Check out **this link** for a comprehensive guide!

You can also do other things like:

- *Italicize* by doing `*this*` and **bold** by doing `**this**`
- Make lists by starting a new line with `-` followed by a space
- Hyperlink using this format: `[whatever text you want](actual link)`

Exercise 1

Try to reproduce this list:

- Everyone should use RMarkdown
- RMarkdown is *very* **cool**
- You can **Google** everything you need to know

How to combine R with your manuscript

So far we saw how to do things that word can do but in RMarkdown instead. That seems a bit silly: why wouldn't we just do all of that in Word? One of the biggest benefits of RMarkdown is that you can imbed Rcodes within your manuscript. That means instead of making a graph in excel, saving the graph, then pasting that graph in Word, you can just have it ready to go in the document, which comes with several advantages:

1. Someone looking at your RMarkdown file can know exactly how you generated the graph and where the data came from
2. You can make minor adjustments to your graph easily by changing the code
3. If the raw data ever changes (e.g. you collected more data and/or there were mistakes in the raw data), you **don't** have to remake the graph because the code is there!

There are two ways to imbed code: inline code and code chunks. Inline codes are generally used for small recalls that you want within your text (e.g. recalling statistical test results and *p*-values.) This is extremely useful for preventing mistakes: instead of copy and pasting results from SPSS and maybe accidentally copying the wrong thing, you are literally asking R to write the results for you. We will get into inline codes a bit later when we discuss how to write APA formatted papers. For now, we will start with **code chunks**, which look like this:

“‘{r}
“‘

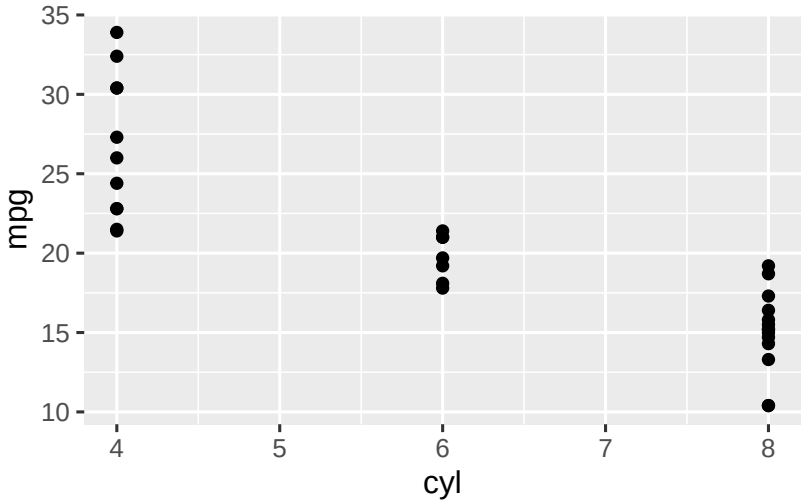
That symbol is NOT an apostrophe; the button is usually under the Esc key. You can also press `Ctrl + Alt + I` to insert a code chunk.

The space between the “`{r} and the closing \``` is where you write your R script, and RMarkdown treats it as such. For example, let's plot something simple from`mtcars‘:

“‘{r}
ggplot(mtcars, aes(x = cyl, y = mpg)) +
geom_point()
“‘

If you press knit, what happens?

```
ggplot(mtcars, aes(x = cyl, y = mpg)) + geom_point()
```



This is not too bad for a report since everyone can see your code, but certainly not viable for a manuscript. We can add `chunk options` to each code chunk to customize how we want each section to be displayed. Here are some common options:

- `echo` = FALSE prevents code from displaying
- `include` = FALSE prevents code AND results from displaying (useful for a code chunk that just loads your packages)
- `fig.width` and `fig.height` lets you adjust your plot's dimensions
- `warning`, `error`, and `message` lets you hide error messages that R might spit out

Now add this chunk option to your code chunk like this and run it again:

"'{r echo = FALSE}