Today I'm diverging into a topic unrelated to Bayesian statistics - R Markdown!

R Markdown is basically an R script but exported in a way that looks fancy . Why do we want our R script to be fancy? Well, first of all it's cool. So there's that. But also, it can be really useful for turning in assignments or showing your code to others in a format that makes the code easier to read. For instance, say you want to combine both text AND code in the same document but don't want to write a reallllly long comment - R Markdown is the solution. By going through the process of making an R Markdown script, you also ensure that *you* know what your code means and can explain it to others, which helps you catch errors and learn more about your code as you go along. Win win!

Let's start with a really easy example. Let's say you want to randomly pull 500 numbers from a normal distribution and then make a histogram of the results. You want the code to be visible to others, but you also want to comment on the resulting histogram for whatever reason. Wow, your code is going to look so cool! Let's do it.

As always, if you have any thoughts on this tutorial, please send any questions or suggestions to heather.e.gaya@gmail.com or find me on twitter @doofgradstudent

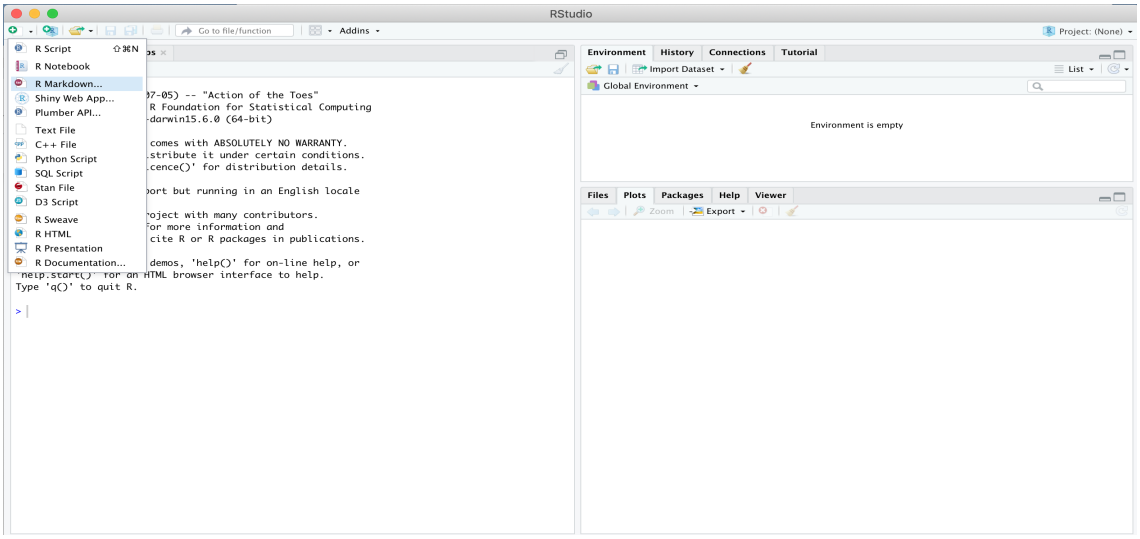
1 Before you Begin

Be sure to install R Markdown before you begin or the "rendering" process (where we turn our lovely R code into an HTML/PDF/DOC file) will not work.

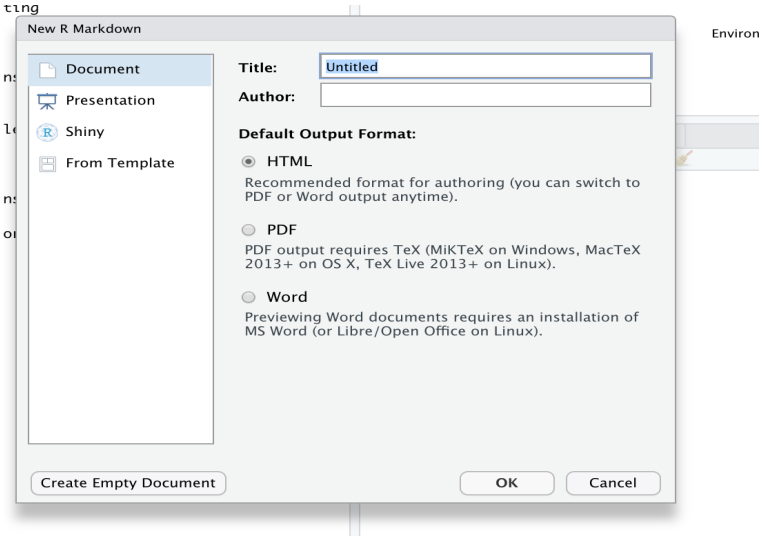
```
1 install.packages("rmarkdown")
```

2 Start the Document

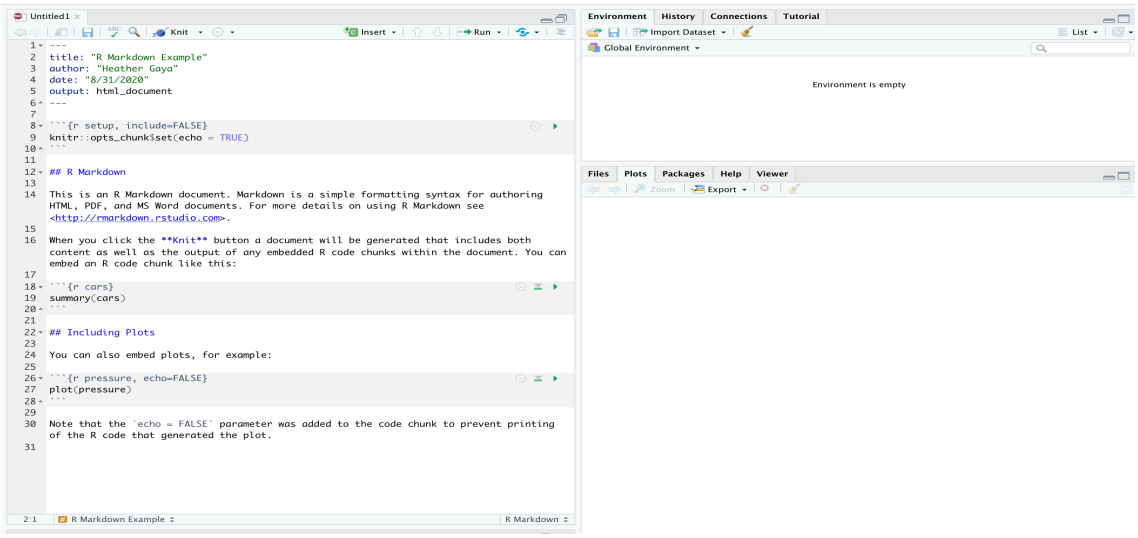
First step, create a new R Markdown script. You can find this by selecting File + New File in the menu or by clicking the "new file" button at the top left of R Studio.



Once there, a window will open asking you give a title to your lovely new R Markdown file. Add a Title and Author, or skip those if you prefer.



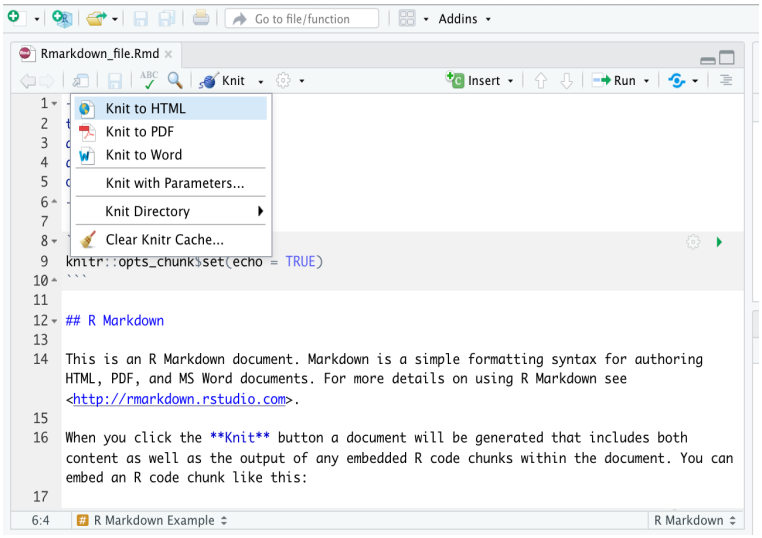
And woo! There's your first new file. Okay, it has nothing useful in it right now but it EXISTS which is pretty special.

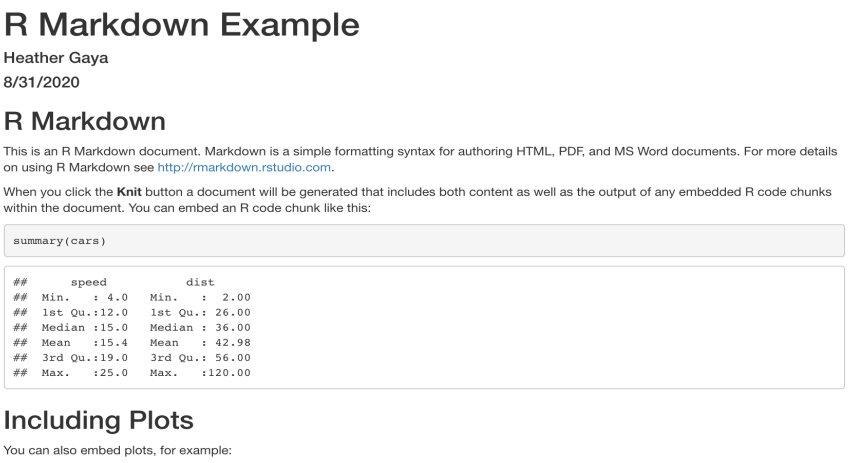


3 What Did I Just Create?

So R assumes, maybe rightfully so, that when you first make an R Markdown file that you need some help. So it fills in a bunch of random info into your file. Quite honestly, this base file is probably a better tutorial than I'm writing right now...

The BEST way to see what R Markdown is actually doing is to click the "Knit" button and see what the output looks like. I like to make my output HTML so that I don't end up with a new file lurking on my computer, but you can use whatever format you wish! If you're submitting the output without the .Rmd file, you probably want a PDF output.





R Markdown Example

Heather Gaya

8/31/2020

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:

So how does the code document relate to the lovely output? Let's go piece by piece.

4 Top of Document

At the top of the R Markdown script, you'll see 3 dashes, followed by some various header slots. There are a LOT of possible header slots with everything from title, author and date to adding a table of contents and sections or changing the font type of your document. There some great info on all the options here: https://bookdown.org/yihui/rmarkdown/html-document.html

But for most uses, title, author, date and output will suffice. If you're ever trying to google this part of the document, this is the "YAML header" which refers to the YAML data language (it stands for "YAML Ain't Markup Language", which is partially a coding joke. Don't worry about it).

Note that the 3 dashes are required both before and after the header to tell R Markdown where your header info starts and ends.

```
---
title: "R Markdown Example"
author: "Heather Gaya"
date: "8/31/2020"
output: html_document
---
```

5 Entering Code

To enter code, either to tell R something or to adjust a setting in R Markdown, you use three backticks (‘‘‘), followed by an r in curly brackets. The r (lower case), tell R Markdown that you will be doing something with R code in the next chunk. Be sure to close the chunk with three more backticks. You can also use the ”add chunk” button at the top of the screen if you prefer.

You can also add options to these code chunks, as seen in the top of the R Markdown file you just created. This line is doing 3 things:

```
‘‘‘{r setup, include=FALSE}
```

First, it tells R Markdown, ”I’m about to start a code chunk here!” Next it names that code chunk ”setup”. This is optional, but a good practice to get into so that your document stays organized. You can rename this whatever you want, though it’s generally a good idea to avoid any special characters or things like that. Finally, ”include = FALSE” tells R Markdown to not print the code below in the final output and also don’t print any results that come out. We’ll see how this differs from ECHO in a minute.

So what exciting code are we giving to R Markdown in this chunk?

```
knitr::opts_chunk$set(echo = TRUE)
```

You’ll notice this is regular R code that you could see in any normal R script. We’re referencing the Knitr package to set a global option for this document. In this case, we’re using a shortcut to tell R Markdown that in the future, unless we say otherwise, we want to see both the output and the results in the finished file. If echo = FALSE we would only see the results, not the code we used to create those results.

Try putting the following into your file, hit ”knit”, and observing the difference:

```
```{r test1}
plot(c(1,1), c(1,2), main = "A Beautiful Plot")
```

```{r test2, echo = FALSE}
plot(c(1,1), c(1,2), main = "Another Beautiful Plot")
```
```

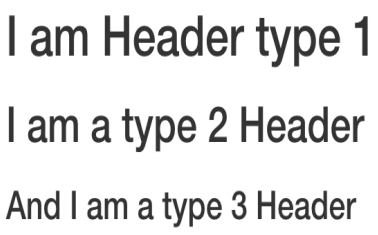
Notice how the first plot shows the code above it, but the second does not. Nifty!

6 Adding Sections, Links, Bold Items and Regular Text

The great thing about R Markdown is that anywhere outside of a code chunk can have normal text without having to comment every line! The downside is, you have to be careful with your pound signs (#) now. In R Markdown, anything that starts with a pound sign suddenly becomes a new section header. The type of header depends on how many pound signs are in front of it.

Compare the following in your document:

```
20 # I am Header type 1
21 ## I am a type 2 Header
22 ### And I am a type 3 Header
23
```



You can also easily make words bold by using two asterisks on either side of a word. In your current document, this is why **Knit** becomes **Knit**. Similarly, you can surround words one asterisk (as in * italicized *) to make them *italicized* or surround them with <and >to create a hyperlink.

7 Let's Graph Something

So now that we know how to make sections, write text, etc., let's try and complete our goal, i.e. randomly pulling 500 numbers from a normal distribution and then making a histogram of the results.

Delete everything in the .Rmd file except your YAML header and your "setup" chunk. Next, let's make a code chunk that creates the 500 numbers and stores them in an object. We'll call the object "numbers". Maybe we also want to comment on why we chose the mean and sd we did, so we'll add a few lines a few lines under our code chunk. Here's our beautiful .Rmd file now:

```
---
title: "R Markdown Example"
author: "Heather Gaya"
date: "8/31/2020"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

#Drawing Our Random Numbers
```{r randnumbs}
numbers <- rnorm(n = 500, mean = 20, sd = 1)
```

Above, we drew 500 numbers from a normal distribution with mean 20 and sd = 1. We chose these numbers, because they were convenient. In other words, no reason. Below, we show a plot of the numbers we pulled.
```

All that's left to do is add a plot! To do that, we add another code chunk like before.

```
---
title: "R Markdown Example"
author: "Heather Gaya"
date: "8/31/2020"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

#Drawing Our Random Numbers
```{r randnumbs}
numbers <- rnorm(n = 500, mean = 20, sd = 1)
```

Above, we drew 500 numbers from a normal distribution with mean 20 and sd = 1. We chose these numbers, because they were convenient. In other words, no reason. Below, we show a plot of the numbers we pulled.

```{r hist}
hist(numbers, main = "500 Draws From a Normal Distribution \nMean = 20, sd = 1")
```
```

And when we knit together the document, here's what we get:

R Markdown Example

Heather Gaya

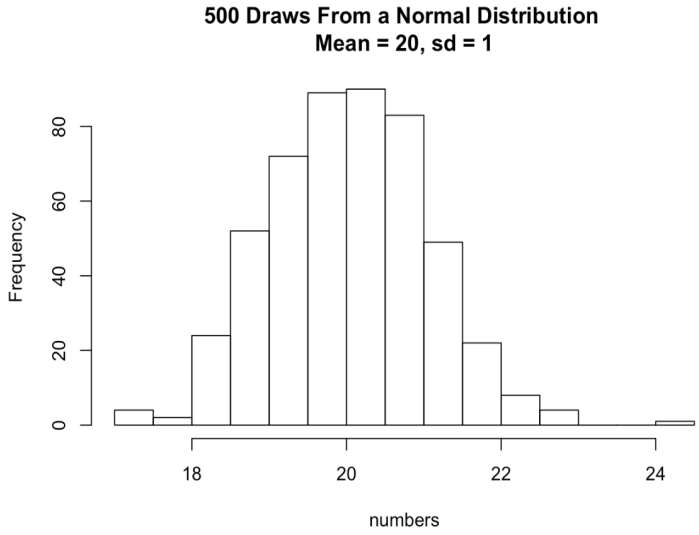
8/31/2020

#Drawing Our Random Numbers

```
numbers <- rnorm(n = 500, mean = 20, sd = 1)
```

Above, we drew 500 numbers from a normal distribution with mean 20 and sd = 1. We chose these numbers, because they were convenient. In other words, no reason. Below, we show a plot of the numbers we pulled.

```
hist(numbers, main = "500 Draws From a Normal Distribution \nMean = 20, sd = 1")
```



Yee-haw!

8 What if I need to Use R Packages?

Let's plot the same information again, this time using an R package so we can practice loading them. Oftentimes we want to use packages but don't want the output to show a message saying we did. Let's see how this works using ggplot. First we'll load the package in one code chunk and be sure to use include = FALSE. Alternatively, if you do want a line showing you loaded the package, but you don't want to show any warning messages, you can load the package using quietly = TRUE.

```
## Using ggplot
```{r loadpackage, include = FALSE}
library(ggplot2)
```

The above code chunk loads ggplot2 but shows no message. The chunk below loads ggplot2
but removes any warning messages if your version of R is old (like mine oops).

```{r loadpackage2}
library(ggplot2, quietly = T)
```
```

In our next chunk of code, which we do want visible, we can plot our graph:

```
## Using ggplot
```{r loadpackage, include = FALSE}
library(ggplot2)
```

The above code chunk loads ggplot2 but shows no message. The chunk below loads ggplot2
but removes any warning messages if your version of R is old (like mine oops).

```{r loadpackage2}
library(ggplot2, quietly = T)
```

Now we can plot our (very exciting) data using ggplot.
```{r plottyplot}
numbersdf <- data.frame(numbers = numbers)
ggplot(data = numbersdf)+
  geom_histogram(aes(x = numbers), binwidth = .5)+
  theme_classic()

```
```

And then finally hit "knit" to see the final output!

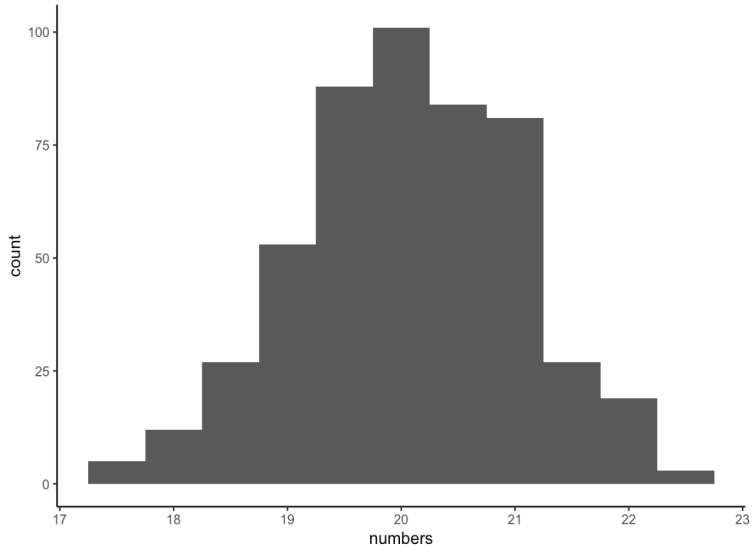
Using ggplot

The above code chunk loads ggplot2 but shows no message. The chunk below loads ggplot2 but removes any warning messages if your version of R is old (like mine oops).

```
library(ggplot2, quietly = T)
```

Now we can plot our (very exciting) data using ggplot.

```
numbersdf <- data.frame(numbers = numbers)
ggplot(data = numbersdf)+
  geom_histogram(aes(x = numbers), binwidth = .5)+
  theme_classic()
```



9 Can I Run my Code Without "Knitting"?

You can always run your code normally in R if you ever want to see what it does! The little green arrow that appears on each code chunk will let you run that chunk on its own, or you can run all the chunks together using the menu at the top of the .Rmd file.

Good luck!