Mastering Your Coding Environment

Mitchell Stares

v1.0

Contents

1 Using your notebook efficiently

R notebook/markdown files are a fantastic solution for working with R scripts. Compared to a normal R script it allows room to write in plain text without needing a whole lot of hashtags in front of sentances. For most, this is where the use of R markdown documents ends. However, there are a number of extra uses for markdown documents that may increase your efficiency and skills within the R environment.

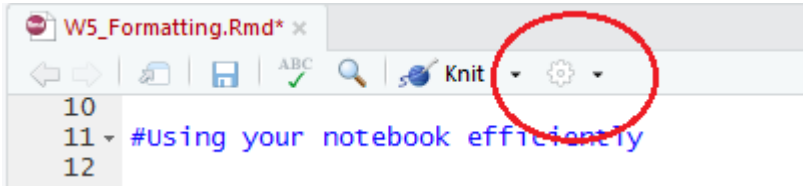
One thing you may have noticed so far is all of your output from coding chunks is placed below the code chunk itself. This is useful, particularly for statistical analysis but can be irritating for graphing.

````
```{r}
iris.scatter <- ggplot(iris, aes(x=Sepal.Length,y=Petal.Length)) +
  geom_point()

iris.scatter
```
````

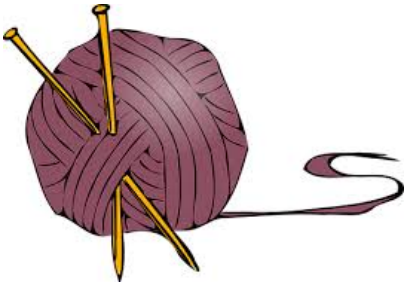
This can be changed so all chunk output is output down in the console and graphs are output in the **plot** window (same window as packages and help). I personally prefer this as it keeps the script environment cleaner during use.

To do this, simply select the settings gear at the top of your script window (next to knit/preview)



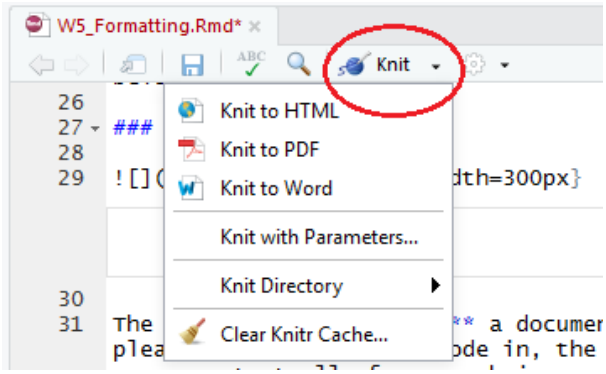
In this dropdown window, select **"chunk output in console"** to change, or keep it as **"chunk output inline"** to keep the preview of your output and graphs below the chunks.

1.1 Knitting



The process of **knitting** a document is one of the main benefits behind using a markdown style document. While markdown documents are aesthetically pleasing environment to code in, the ability to export your work to a different file type is very handy. The use of code chunks and non-coding regions means we can output all of our work in our markdown file to a pdf, word or html with the click of a button (and sometimes some formatting).

To export your document to another file type, simply click the knit button to drop down the menu and click your preferred file type.



2 Formatting Your Code

Formatting our code so it is easy to read, follow and reproduce is an important part of open-access science and helpful when sharing our code to collaborators or friends. It can also be useful when we look back at a past script to remember how to do something.

Before we start explaining some of this, download and look at the R Markdown Reference Guide.

This will help if ever you wish to produce a word, pdf or html document directly from your markdown document (which is how I made these tutorials).

3 Headings

When working in a R markdown document (R markdown or R notebook) there are a number of ways to change the appearance of text within these non-coding regions.

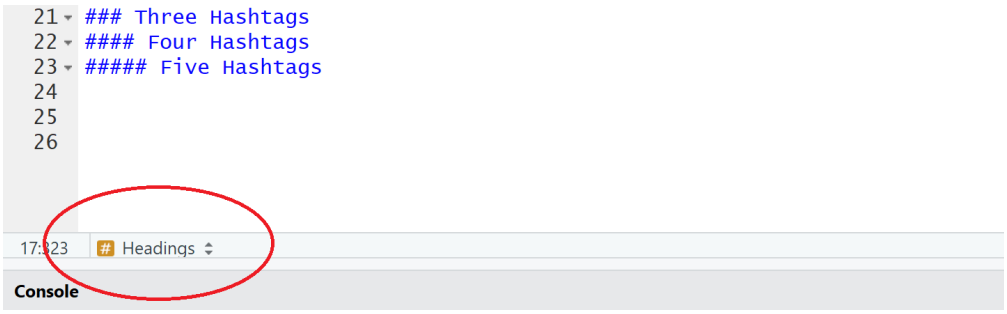
The easiest way to make your code more readable and navigatable (is that even a word?) is to place some headings before different sections of your code. This is simply done by placing a # (hashtag) before your heading on its own line. e.g. # Heading. Because this is not on its own line, it will not default to a heading.

```
# One Hashtag
## Two Hashtags
### Three Hashtags
#### Four Hashtags
##### Five Hashtags
```

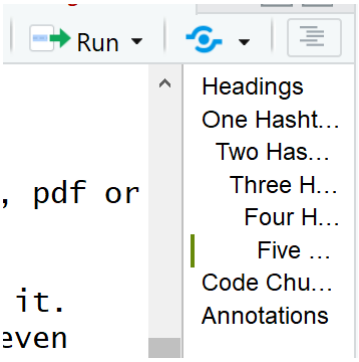
Once knitted, these will appear in a large font text like so:

Even without these **knitted** into another document format, these are still useful in your document.

These will not only help in dividing up your document, but will assist in navigating it. Navigating code can be painful at the best of times but when there are hundreds (or even thousands) of lines of code, it's pure hell. The navigation tools in R studio are very useful when this happens.



Clicking this navigation button (where is says Headings) will bring up a list of all the **# headings** you have throughout your document, along with all the code chunks. The second method of navigating is by using the **document outline** window located at the edge of your window near the **run** button.



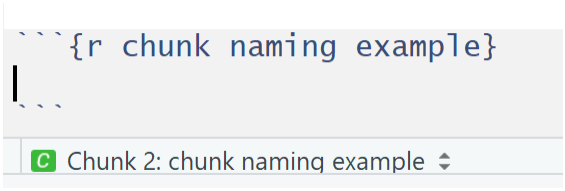
The document outline button is the one that looks like a list (next to the "pokeball" button). As you can see, this will list all your headings in the document, but wont list any of you coding chunks.

Which brings us to the next helpful tip.

4 Code Chunks

Along with navigating by headings, we can also navigate by code chunks. Within the navigation tab you will probably see all the code chunks listed "Chunk 1", "Chunk 2", "Chunk 3" etc. To make this easier, we can actually name our chunks to help with this navigation.

This is done by writing the name we want for our chunk directly after the **r** within our wing brackets **{ }**, like so: **{r chunk name}**. This will display this in the navigation tab, along with just making it easier to see what the chunk might be for.



In addition to this, we can also use "headings" in our coding chunks to help organise sections of chunks. This is done by placing anything inside four hashtags. e.g. #### **ANOVA** ####. Once you place 4 on either side of something inside a chunk, it will appear in the navigation tab. **Personally, I dont use this, but I will use hashtag like headings or a row of -'s to separate parts of code**.

```
#### Data Modification ####

data <- read.csv("filename.csv")

#### ANOVA ####

data_anova <- aov(something.important ~ something.else, data=data)

#### Linear Regression ####

data_lm <- lm(something.important ~ something.continuous, data=data)
```

The above example is incredibly over the top when it comes to headings and organising but it does assist in reading and following what the author has done.

5 Annotations

Annotating your code is crazy important for a number of reasons. The main reason is for you personally when you look back on what you have coded. It helps to explain in detail what a line, chunk or even section of code is trying to accomplish.
Behind every great R statistician is a library of annotated code they will "repurpose". Some of the best resources you can have for working with R (or any coding language for that fact) is previous work that you have carefully annotated and explained in your own words.

This is also helpful for other people who read you code. Explaining what a line of code is doing can be useful for other who are looking to adapt your work to their own. It also helps when showing the code to a collaborator, supervisor or someone else who might be marking/critising your work :)

The easiest ways to annotate code is inside the chunk using #'s.

```
#### Data Modification ####

data <- read.csv("filename.csv")
# reading in the data file

#### ANOVA ####

data_anova <- aov(something.important ~ something.else, data=data)
# constructing an anova object with something.important as the Y variable
# and something.else as the x variable

#### Linear Regression ####

data_lm <- lm(something.important ~ something.continuous, data=data)
# constructing a linear model object using the lm() command.
```

This is pretty excruciating detail but it helps. Most people you would be sending code to should be fairly familiar with basic functions so you would adapt your annotations accordingly. Any code I am producing for a learning resource will be annotated within an inch of its life so that way the information is there if it is needed.