MA256 Lesson 2 - R Studio/Markdown Tips (AY24-1)

R Markdown

- **rmd** files operate much like `R` files except that they use **chunks** to execute lines of code.
- outside of the **chunks**, we can use `LaTex` functionality to make our documents more visually appealing (see below for more 'mathy' stuff).
- I **DO NOT** require that you use `RMarkdown` files over `R Scripts`, but the daily products I provide will be using `RMarkdown`.

Starting out

- You should always start from a **clean** RMarkdown file.
  - Open up a new file (or delete everything in an already existing file)
  - Save it as something different (in a folder that makes sense)
  - Sweep your **Global Environment** with the broom icon in the upper right.

Shortcuts

- `alt|enter`: runs current line
- `ctrl|enter`: runs current line **and** tabs down to next line
- `ctrl|alt|i`: inserts new code chunk
- `ctrl|shift|c`: comments line of code (or selection)
- `ctrl|-`: zoom out
- `ctrl|shift|+` : zoom in

R Basics

```
x = 1 # assigns value to variable
y <- 2 # another way to do leftward assignment
3 -> z # rightward assignment (NOT GOOD PRACTICE)
rm(x) # remove a variable from the global environment
rm(y,z) # remove several variables
```

Vectors

```
# a vector in R (an ordered list)
list1 = c(1, 2, 3, 2)
list1
```

```
## [1] 1 2 3 2
```

```
list2 = c("a", "b", "c", "a")
list2
```

```
## [1] "a" "b" "c" "a"
```

```r
class(list1)
```

```
## [1] "numeric"
```

```r
class(list2)
```

```
## [1] "character"
```

Data frames

```r
# R likes to operate on vectors
df <- data.frame(list1, list2)
```

```r
# indexing in r
df$list1
```

```
## [1] 1 2 3 2
```

```r
df$list2
```

```
## [1] "a" "b" "c" "a"
```

```r
df[1, ] # returns the first row
```

```
##   list1 list2
## 1     1     a
```

```r
df[ ,1] # returns the first column
```

```
## [1] 1 2 3 2
```

```r
# functions in r
# format:    function(arg1, arg2, arg3)
mean(df$list1)
```

```
## [1] 2
```

```r
 # to get help on a function
```

Tidyverse

- `%>%` : pipe operator
- used in `tidyverse` package to simplify nested commands
- `ctrl|shift|m`: inserts pipe operator

Packages

- A package is a collection of functions.
- Packages only need to be installed once, but we run the `library(insert_package_name_here)` at the start of each file.

```
# Installing Packages
#install.packages("tidyverse")
library(tidyverse)
```

Pipe Operator

```
# Two ways of doing the same thing:
df %>% filter(list1 > 1)
```

```
##   list1 list2
## 1     2     b
## 2     3     c
## 3     2     a
```

```
filter(df, list1 > 1)
```

```
##   list1 list2
## 1     2     b
## 2     3     c
## 3     2     a
```

```
# Notice how this function has the same output
```

```
# %>% becomes useful for multiple operations on an object:

df %>%
  filter(list1 > 1) %>%
  summarize(mean = mean(list1))
```

```
##       mean
## 1 2.333333
```

```
summarize(filter(df,list1 > 1), mean = mean(list1))
```

```
##       mean
## 1 2.333333
```

```
# Which method is more intuitive?
```

Navigation

- You can jump to chunks and titles by clicking at the bottom of this document.
- Math operators in R: https://www.datamentor.io/r-programming/operator/
- Clear variables with broom icon in Environment Tab (top right window)
- The Plots Tab (bottom right window) and Console (bottom left window) also have the broom option.
- You can run current chunk by clicking the green arrow at the top right of the chunk, by using `ctrl + alt + c`, or `ctrl|shift|enter`.
- You can run the entire document up until your current chunk by clicking the down arrow at the top right of chunk.

More Stuff

```
x = 5
class(x) # x is numeric type variable
```

```
## [1] "numeric"
```

```
y = c(5,6,7) # saves the list of numbers to the variable x
class(y)
```

```
## [1] "numeric"
```

```
y = c(5,2,"cat") # all numbers are converted to strings so that all list elements are the same type
class(y) # now y is a character (aka "string") type variable
```

```
## [1] "character"
```

```
y[1] # access the first entry of y.  "5" is a string (aka "character" type)
```

```
## [1] "5"
```

```
class(y[1])
```

```
## [1] "character"
```

```
x[1] # access the first entry of z.   5 is a numbe r (aka "numeric" type)
```

```
## [1] 5
```

```
class(x[1])
```

```
## [1] "numeric"
```

```r
y = c(5,6,7)

x+y
```

```
## [1] 10 11 12
```

```r
z = c(1,2,3)
x + z
```

```
## [1] 6 7 8
```

```r
x*y
```

```
## [1] 25 30 35
```

```r
# Remove variables with rm() function
rm(x) # remove one variable
rm(y,z) # remove multiple variables

# Getting help on a function
# ?rm   # ?function_name to get help on a function
# Google is also a great reference for troubleshooting R
# You can copy error messages into Google and often find a solution
```

Mathy stuff

We can also use LaTeX code to provide some mathy symbols:

- We can make Greek letters, such as β, γ, σ, and λ.
- I can make a fraction, $\frac{a}{b}$, using in-line symbols.
- I can also make the square root of a fraction of squared and cubed terms using `math mode`:

$$\sqrt{\frac{a^2}{b^3}}$$

(optional) For more examples for LaTeX mathematics take a look at these sites: https://en.wikibooks.org/wiki/LaTeX/Mathematics

https://www.cmor-faculty.rice.edu/~heinken/latex/symbols.pdf