Class 1 - R Syntax

Learning to code!

For those of whom this is your first time programming, welcome! To me, programming is a tool to get the computer to do what you want (often to learn from data). I see it as an exploration tool.

This class will be as much a learning experience for me as it will be for you. I'll do my best to teach what I see as the fundamentals as clearly as possible, but let me know if there's anything that you're unsure about - and ask questions!

Getting familiar with RStudio

Four panes! Each do something different. You can move them around, but here's how mine are:

1) Top-left -> "source" - this is what you write in and save to your computer. Think of this like your Word doc!
2) Top-right -> "environment" - this is where you can see what all of your objects are. You can click on them and explore them.
3) Bottom-left -> "console" - this shows a running log of what you've run. You can write forever up in the "source" pane, then once you run something, you see it down below.
4) Bottom-right -> This is sort of the miscellaneous area, but very useful! It has the help pane, "viewer" for plots, and your files (so you can see what files you have in your local folder).

To get familiar with RStudio, we'll first change its appearance: Tools -> Global Options -> Appearance. change whatever you'd like! I use Pastel on Dark as my editor theme - it saves your eyes :)

Reading resources: https://bookdown.org/ndphillips/YaRrr/the-four-rstudio-windows.html

Object assignment

```
"Hello world"
```

```
## [1] "Hello world"
```

```
obj1 <- "This is a string"

## what class is this object?
class(obj1)
```

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

```
obj2 <- 9999
class(obj2)
```

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

```r
obj3 <- iris ## iris is a built-in dataset within R
class(iris)
```

```
## [1] "data.frame"
```

```r
## On your own -> try opening our new iris object in the Enviroment!

obj4 <- c(1, 2, 3, 4, 5, 1000) ## a list
obj4 == c(1:5, 1000)
```

```
## [1] TRUE TRUE TRUE TRUE TRUE TRUE
```

```r
obj4[6] ## this is called an index
```

```
## [1] 1000
```

```r
which(obj4 == 1000) ## getting the index
```

```
## [1] 6
```

```r
## With your teams: make a new object called "team_names" and assign it the first names of everyone on

## On your own -> find and understand the main data types and data structures in R. Try some new ones b
```

Messing with objects

```r
1 + 1
```

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

```r
-500 + obj2
```

```
## [1] 9499
```

```r
#"put us" + "together!" ## All together - let's fix this!

head(obj3)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
### HERE COMES THE MOST DIFFICULT BUT MOST IMPORTANT PART!! ###
###-------- navigating matrices -> [row, column] -----------###
obj3[1, 1]
```

```
## [1] 5.1
```

```
obj3[1, ]
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
```

```
obj3[, 1]
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4
##  [18] 5.1 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5
##  [35] 4.9 5.0 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0
##  [52] 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8
##  [69] 6.2 5.6 5.9 6.1 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4
##  [86] 6.0 6.7 6.3 5.6 5.5 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8
## [103] 7.1 6.3 6.5 7.6 4.9 7.3 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7
## [120] 6.0 6.9 5.6 7.7 6.3 6.7 7.2 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7
## [137] 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8 6.7 6.7 6.3 6.5 6.2 5.9
```

```
## easier to refer to columns by their name, not number
obj3$Species
```

```
##   [1] setosa     setosa     setosa     setosa     setosa     setosa
##   [7] setosa     setosa     setosa     setosa     setosa     setosa
##  [13] setosa     setosa     setosa     setosa     setosa     setosa
##  [19] setosa     setosa     setosa     setosa     setosa     setosa
##  [25] setosa     setosa     setosa     setosa     setosa     setosa
##  [31] setosa     setosa     setosa     setosa     setosa     setosa
##  [37] setosa     setosa     setosa     setosa     setosa     setosa
##  [43] setosa     setosa     setosa     setosa     setosa     setosa
##  [49] setosa     setosa     versicolor versicolor versicolor versicolor
##  [55] versicolor versicolor versicolor versicolor versicolor versicolor
##  [61] versicolor versicolor versicolor versicolor versicolor versicolor
##  [67] versicolor versicolor versicolor versicolor versicolor versicolor
##  [73] versicolor versicolor versicolor versicolor versicolor versicolor
##  [79] versicolor versicolor versicolor versicolor versicolor versicolor
##  [85] versicolor versicolor versicolor versicolor versicolor versicolor
##  [91] versicolor versicolor versicolor versicolor versicolor versicolor
##  [97] versicolor versicolor versicolor versicolor virginica  virginica
## [103] virginica  virginica  virginica  virginica  virginica  virginica
## [109] virginica  virginica  virginica  virginica  virginica  virginica
## [115] virginica  virginica  virginica  virginica  virginica  virginica
## [121] virginica  virginica  virginica  virginica  virginica  virginica
## [127] virginica  virginica  virginica  virginica  virginica  virginica
## [133] virginica  virginica  virginica  virginica  virginica  virginica
## [139] virginica  virginica  virginica  virginica  virginica  virginica
## [145] virginica  virginica  virginica  virginica  virginica  virginica
## Levels: setosa versicolor virginica
```

```r
unique(obj3$Species)
```

```
## [1] setosa     versicolor virginica
## Levels: setosa versicolor virginica
```

```r
obj3$Sepal.Length
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4
##  [18] 5.1 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5
##  [35] 4.9 5.0 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0
##  [52] 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8
##  [69] 6.2 5.6 5.9 6.1 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4
##  [86] 6.0 6.7 6.3 5.6 5.5 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8
## [103] 7.1 6.3 6.5 7.6 4.9 7.3 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7
## [120] 6.0 6.9 5.6 7.7 6.3 6.7 7.2 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7
## [137] 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8 6.7 6.7 6.3 6.5 6.2 5.9
```

```r
summary(obj3$Sepal.Length)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.300   5.100   5.800   5.843   6.400   7.900
```

```r
## With your teams - what are three ways to find the max and min Petal Length?

## THE MOST IMPORTANT THING YOU NEED TO KNOW!!! "conditional subsetting"
just_sepal <- obj3$Sepal.Length
just_short_sepal <- obj3[obj3$Sepal.Length <= 5.1, ] ## THIS IS GOLD

## With your teams - get a subset of your data (all columns) where the Species is setosa and call it "s
```

How to read documentation

We'll google the examples `paste` function, both using the built-in functionality (see below) and Google.

Fun and useful R tips!

- Type and run `?sum` or `?head` or `?paste` in the console. You can do this for any function, command, or package to learn more. It's like built-in Google!
- Click into the console and press the up arrow. What happened? This is useful for rerunning lines of code, or if you want to make a slight modification.
- Tabbing - make an object called `wow_what_a_fun_object`. Now, in either the source or console, start typing `wow_` and press tab. It auto-completes!
- Comments (`#`) will save your life - you can document what you've done right in your code. You'll hear programmers talk about "well-commented code" - it makes it easier to follow along with what you've done, which will save you time, too. I also keep track of findings as I go in comments.
- Well-formatted code has lots of spaces - you can look documentation on style guides if you'd like. Everything I write here will be well styled, so just try and mimic.