Lab 1: Introduction to R and RStudio
STAT 630, Fall 2020

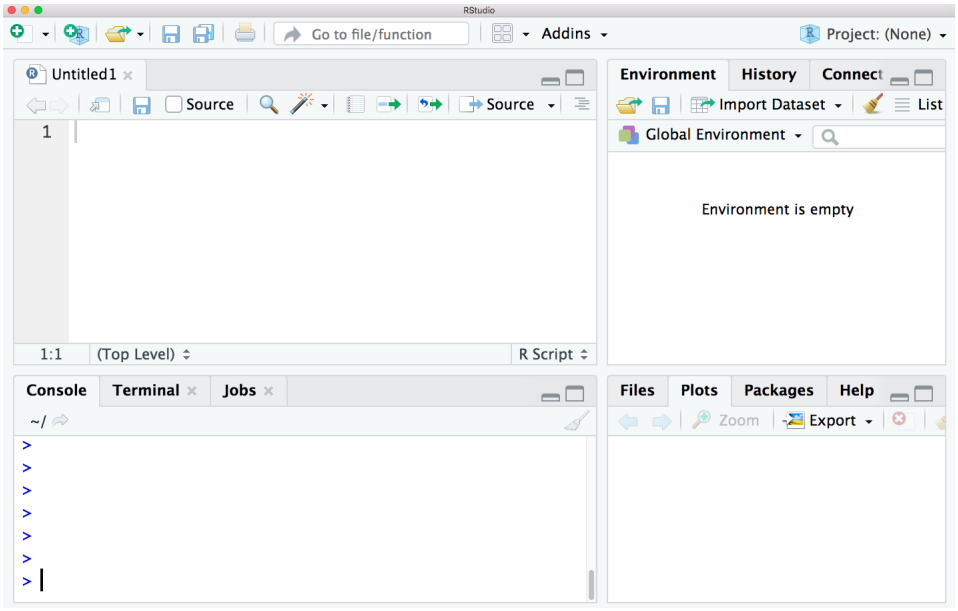
1 Installing R and RStudio

To download the latest version of R go here: `https://cloud.r-project.org/`
Click on the "Download R" link for your operating system (Windows, Mac) and follow the instructions for installation.

To download the latest version of RStudio go here:
`https://rstudio.com/products/rstudio/download/`
Click on the button to download "RStudio Desktop Open Source License" (free version). Next click on the link to download RStudio for your operating system (Windows, Mac) and follow the instructions for installation.

2 Getting Started with RStudio

Go ahead and open RStudio, what you see should look like the screenshot below. For clarification: R is the name of the programming language and RStudio is a convenient interface.



RStudio is divided into four panels:

- Top left panel: This is the code editor, this is where you write your code. To create a new R script go to File → New File → R Script. Make sure to save your R scripts so that you can access them later.

- Bottom left panel: This is the console, this is where you run your code. You can either copy and paste code from an R script to the console, or use the shortcut Ctrl+Enter (Windows) or Command+Enter (Mac) to run a line of code.
- Bottom right panel: Any plots you create will show up in this pane. This pane also contains the help menu, where you can read documentation about R functions.
- Top right panel: This contains the Environment and History panes. The Environment pane shows everything that is currently in your workspace (e.g., objects you create or data sets you have loaded for analysis). The History pane provides a history of all previous commands you have run in the console.

3 Using R as a calculator

You can use R for basic calculations (addition, subtraction, multiplication, exponentiation, taking square roots). Try typing a few simple computations into the console. Press the enter key to run each command. After enough practice you might end up preferring R to your calculator.

```
2 + 2
```

```
## [1] 4
```

```
2 * 3
```

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

```
2^5
```

```
## [1] 32
```

```
sqrt(2)
```

```
## [1] 1.414214
```

```
7.2 + 2 * 2.1 / sqrt(101)
```

```
## [1] 7.617916
```

If you want to edit a previous command in the console, press the up arrow key. For instance, I may want alter the last computation:

```
7.2 - 2 * 2.1 / sqrt(101)
```

```
## [1] 6.782084
```

4 Variable Assignment

The <- symbol is called the assignment operator. It assigns values to variables. Note that the = symbol can also be used for assignment. For example, here is an example of using R to compute the quadratic formula for values of a, b, and c:

```
# find solutions to the equation 5x^2 + 6x + 1 = 0
# use quadratic formula
a <- 5
b <- 6
c <- 1
(-b + sqrt(b^2 - 4*a*c)) / (2*a)

## [1] -0.2

(-b - sqrt(b^2 - 4*a*c)) / (2*a)

## [1] -1
```

The # sign is used to write comments. Anything to the right of # is not evaluated in the console. Comments are useful when sharing code with others (or to help me understand my code in the future!).

5 Vectors

A numeric vector is just a collection of numbers. To create a vector use `c()`, which is short for combine.

```
x <- c(1, 2, 3, 47)
x

## [1]  1  2  3 47

y <- c(1, -1, 2, -2)
y

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

We can add and multiply vectors. We can also multiply or divide a vector by a scalar. Operations are performed element-wise.

```
x + y

## [1]  2  1  5 45

x * y

## [1]   1  -2   6 -94

2*x # multiplies each element by 2

## [1]  2  4  6 94

x/2 # divides each element by 2

## [1]  0.5  1.0  1.5 23.5

x^2 # squares each element of the vector

## [1]    1    4    9 2209
```

5.1 Sequences

Some ways to create vectors containing sequences of numbers in R:

```
x <-  1:10
x

##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
y <- seq(2, 20, by=2)
y
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20
```

```r
z <- seq(0, 1, by=0.1)
z
```

```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

Use `rep()` to repeat elements in a vector:

```r
rep(1,10) # repeats 1 ten times
```

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

```r
x <- c(rep(2,3), rep(3,4))
x
```

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

5.2 Indexing

To extract specific elements of a vector use brackets `[]`. The number in the bracket is the index, that is, the position of the element within the vector.

```r
x <- c(2, 7, 15, 21, 47, 100, 101)
x[1]
```

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

```r
x[4]
```

```
## [1] 21
```

```r
x[c(1,4)]
```

```
## [1]  2 21
```

```r
x[1:3]
```

```
## [1]  2  7 15
```

```r
x[-1] # drop first element
```

```
## [1]   7  15  21  47 100 101
```

```r
x[-c(1,4)] # drop first and fourth element
```

```
## [1]   7  15  47 100 101
```

5.3 Vector functions

R provides some easy-to-use functions for doing common operations such as computing the sum or length of a vector. Below is a demonstration of some useful functions:

```
x <- c(2, 7, 15, 21, 47, 100, 101)
length(x)

## [1] 7

sum(x)

## [1] 293

min(x)

## [1] 2

max(x)

## [1] 101

mean(x)

## [1] 41.85714

median(x)

## [1] 21
```

To learn more about a function use the help menu. To access the help menu from the console use the `help()` function. For example, enter the following command:

```
help(mean)
```

5.4 Data types

There are four common data types for vectors: numeric (also called double), integer, character, and logical. Some examples:

```
x <- c(1, 0.5) # numeric
x <- 1:10 # integer
x <- c("a", "b", "c") # character
x <- c(TRUE, FALSE) # logical
```

You can use `class()` to check the data type.

```
letters

##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
## [20] "t" "u" "v" "w" "x" "y" "z"

class(letters)

## [1] "character"
```

When different data types are mixed in a vector, R implicily converts all elements to the same type.

```
x <- c("a", 2)
x

## [1] "a" "2"

class(x)

## [1] "character"
```

6 Data Frames

Data sets in R are represented as objects called data frames. The columns of a data frame represent the variables in the data set, and the rows of a data frame are the specific cases or observations. For example, lets start by examining a data frame called `mtcars`, which is already stored in R. This data set was from the 1974 *Motor Trend* magazine, and contains variables (columns) on automobile design and performance for 32 different automobile models (rows).

```
head(mtcars) # preview first several rows of data frame

##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

You can also type `mtcars` into the console, and it will print out the entire data set. But for larger data sets this is not advisable. To learn more about this data, use the help menu by typing `help(mtcars)` into the console.

Below are some useful functions for exploring a data frame in R:

```
nrow(mtcars) # number of rows

## [1] 32

ncol(mtcars) # number of columns

## [1] 11

dim(mtcars) # dimension

## [1] 32 11

names(mtcars) # names of the columns (variables)

##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
## [11] "carb"

row.names(mtcars) # names of the rows (car models)

##  [1] "Mazda RX4"           "Mazda RX4 Wag"       "Datsun 710"
##  [4] "Hornet 4 Drive"      "Hornet Sportabout"   "Valiant"
##  [7] "Duster 360"          "Merc 240D"           "Merc 230"
## [10] "Merc 280"            "Merc 280C"           "Merc 450SE"
## [13] "Merc 450SL"          "Merc 450SLC"         "Cadillac Fleetwood"
## [16] "Lincoln Continental" "Chrysler Imperial"   "Fiat 128"
## [19] "Honda Civic"         "Toyota Corolla"      "Toyota Corona"
## [22] "Dodge Challenger"    "AMC Javelin"         "Camaro Z28"
## [25] "Pontiac Firebird"    "Fiat X1-9"           "Porsche 914-2"
## [28] "Lotus Europa"        "Ford Pantera L"      "Ferrari Dino"
## [31] "Maserati Bora"       "Volvo 142E"
```

To access specific columns, or variables, use the `$` operator. For example, we can extract the `mpg` column (miles per gallon) from `mtcars`:

```
mtcars$mpg

##  [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4
## [16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7
## [31] 15.0 21.4
```

Notice that when we access a column this way, we are actually extracting a vector. So we can use some of the vector functions, previously mentioned, to explore.

```r
max(mtcars$mpg)
```

```
## [1] 33.9
```

```r
min(mtcars$mpg)
```

```
## [1] 10.4
```

```r
mean(mtcars$mpg)
```

```
## [1] 20.09062
```

```r
median(mtcars$mpg)
```

```
## [1] 19.2
```

We can also use indexing to extract specific columns or rows of a data frame.

```r
mtcars[, 1] # extract first column
```

```
##  [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4
## [16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7
## [31] 15.0 21.4
```

```r
mtcars[1, ] # extract first row
```

```
##           mpg cyl disp  hp drat   wt  qsec vs am gear carb
## Mazda RX4  21   6  160 110  3.9 2.62 16.46  0  1    4    4
```

```r
mtcars["Datsun 710", ] # extract row for Datsun 710
```

```
##             mpg cyl disp hp drat   wt  qsec vs am gear carb
## Datsun 710 22.8   4  108 93 3.85 2.32 18.61  1  1    4    1
```

```r
mtcars[2, 3] # extract element in the second row and third column
```

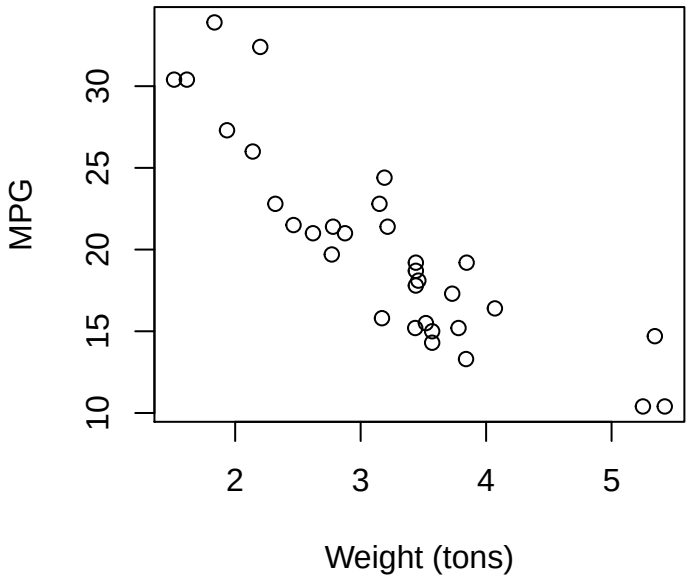
```
## [1] 160
```

```r
mtcars[2:3, ] # extract second and third row
```

```
##                mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710    22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
```

The `plot()` command makes a scatter plot of two variables. Below is a scatter plot with the weight of the car on the x-axis and mileage on the y-axis.

```
plot(mtcars$wt, mtcars$mpg, xlab='Weight (tons)', ylab='MPG')
```



Based on this scatter plot we can see that there is a decreasing and somewhat linear relationship between weight and mileage. That is, as weight increases the mileage decreases on average for car models in this data set, which makes sense.

That was a lot of information if this is your first time using R. Congratulations! Take a deep breath, you will have the rest of the semester to get comfortable with R and RStudio. We will review this material in future labs, and you will see the same commands and functions repeatedly used in different contexts.

Practice Problems (not to be collected)
Note that exercises 4 through 6 will use the `mtcars` data frame.

Exercise 1. Use R to find the sum of numbers from 1 to 1000.

Exercise 2. Use `seq()` to express the sequence $2, 4, 6, \cdots, 200$ as a vector in R.

Exercise 3. Use `rep()` to create a character vector with the letter "a" repeated 10 times, followed by the letter "b" repeated 20 times.

Exercise 4. Find the min, max, mean, and median of the car weight variable (`wt`).

Exercise 5. Make a plot of `mpg` versus `cyl`. Label the y-axis "Miles per gallon" and the x-axis "Number of cylinders". Describe the association between the two variables.

Exercise 6. We can use `which.min()` and `which.max()` to find the index of the minimum and maximum element of an R vector. For example, look at the following simple example:

```
x <- c(2, 7, 100, -1, 5)
which.max(x)

## [1] 3

which.min(x)

## [1] 4
```

The following command will extract the row of the `mtcars` data frame corresponding to the car with the minimum `mpg`.

```
mtcars[which.min(mtcars$mpg), ]

##                     mpg cyl disp  hp drat   wt  qsec vs am gear carb
## Cadillac Fleetwood 10.4   8  472 205 2.93 5.25 17.98  0  0    3    4
```

Similarly, we can extract the row corresponding to the car with the maximum `mpg`:

```
mtcars[which.max(mtcars$mpg), ]

##                 mpg cyl disp hp drat    wt qsec vs am gear carb
## Toyota Corolla 33.9   4 71.1 65 4.22 1.835 19.9  1  1    4    1
```

Using this technique, which car model in the `mtcars` data frame has the minimum weight? Which car model has the maximum weight? Recall that weight is the column named `wt`.