1 Class 1 - June 1

1.1 Basics of R Syntax

1. Write a two line R script. Make the first line a comment and the second line a simple addition problem. Run the script to ensure that you get the expected result.

2. Add a third line to the script from above. This line should be another simple addition problem. Run each of the three lines individually, then again as an entire script.

3. Explain why `[1]` appears after running `4+4`.

4. Create an object called height and assign your height in inches as the value.

5. Convert the height object to centimeters ($1\ in. \approx 2.54\ cm$). Assign the result to the object `height`.

6. Run `height`. Why did this print a value to the screen, when running the above command did not?

1.2 R Markdown

1. Create a pdf in R Markdown. It should contain a brief explanation of your proposed final project idea. It should also contain two R chunks. The first chunk should assign a variable called `normDist`, which contains 1000 numbers drawn from a normal distribution. The second chunk should contain a graph of those numbers.

2 Class 2 - June 2

2.1 Data Types, Vectors, and Subsets

I recommend that you try to answer the questions without entering anything into the computer. Then, go back through and double-check your responses. It is good practice to

double-check, even if you are sure of the answer.

1. For each of the following, give the class of the data.

   (a) c(1, 2, 3)
   (b) "Dr. H"
   (c) c(1, 2, 3, "Dr. H")
   (d) x <- c(1, 2, 3, "Dr. H")
   x[1:3]
   (e) True
   (f) mtcars["mpg"]
   (g) mtcars[["mpg"]]

2. Create a vector that contains the numbers 1 to 10. Complete the following and state the end result.

   - multiply the vector by 2
   - add 4 to the vector
   - divide the vector by 10

3. The `mean()` function, unsurprisingly, gives you the mean of a set of values. For example, we can use `mean(mtcars$mpg)` or `mtcars[["mpg"]]` to get the mean `mpg` of the `mtcars` dataframe. Similarly, `sd()` can be used to get the standard deviation of a numeric vector. Subset appropriately to get the following values from the mtcars dataframe:

   (a) the mean and standard deviation `hp`
   (b) the mean and standard deviation `mpg` for cars that have an mpg greater than 25
   (c) the mean and standard deviation `wt` for cars that have a `cyl` value of 4
   (d) the mean and standard deviation `mpg` for cars that have a `cyl` value of 4 and that have a `wt` less than 3

4. Calculate the `mpg` divided by the `wt` for each car.

5. Use vectorized operations to calculate the mean of the vector

`c(21,14,26,18,22,31,26,36`. As a hint, the `length()` function can be used to get the number of elements in a vector. I also recommend that you save each step along the way. Remember, the mean can be calculated using the formula:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

6. Use vectorized operations to calculate the sum of squares, variance, and standard deviation for the same numbers. As a reminder, the formulas are as follows:

$$SS = \sum_{i=1}^{N}(x_i - \bar{x})^2$$

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

$$s = \sqrt{s^2}$$

7. Add a constant to each of the numbers in `c(21,14,26,18,22,31,26,36)`, then calculate the mean, sum of squares, variance, and standard deviation again. Which numbers changed?

8. Create a dataframe with the following columns, in this order, from `mtcars`: `mpg, hp, wt, cyl`. How does this relate to the order that they were originally in?

9. Turn `mtcars$cyl` into a factor. Multiply the resulting values by 2. What is the output? Why did you get that output?

10. Explain the difference between an atomic vector and a list. What is the benefit to using each?

3 Class 3 - June 3

3.1 Loops

1. Print the multiplication table (from 1 to 10) for the number 13.

2. Given `c(10,13,18,31)`, write a program to subtract the mean from each number using a for loop.

3. Given `c(18,62,41,5,16,62)`, write a for loop that squares each of the even numbers. (hint: the `%%` operator will likely be useful)

4. Given `c(1:50`, multiply each number by 6, then print all of the numbers divisible by both 2 and 3.

3.2 Control Flow

1. Imran Ghory wrote a classic problem to test whether a job applicant can program using if-else logic. It is called FizzBuzz, and goes like this:

   > Write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz".

2. Add the positive integers together until you reach a number the is larger than 30. Print the largest integer that you added.

3. Beginning with the number one, iterate using a while loop until you find the first number divisible by 3, 4, and 5. (hint: the `!=` operator will likely be useful)