PSTAT 10 Worksheet 1

Due 6/28/22 11:59pm

Using RMarkdown

Your TA will introduce R Markdown's basic formatting syntax along with how to run R code in an R Markdown document. For PSTAT 10, you should "knit" your result **to pdf** to upload to Gradescope. To knit to pdf, you must install the `tinytex` package:

```
install.packages('tinytex')
tinytex::install_tinytex()
```

You must also have the following in the metadata header in your RMarkdown document:

```
output: pdf_document
```

Problem 1: The working directory

In my experience, even technologically capable students sometimes struggle with their computer's filesystem. For this class, I highly recommend you to pick a location on your computer in which to store all class files. The best practice is to have a dedicated spot for your code and not to use the desktop or downloads folder.

After selecting a spot, set it as your R *working directory*. The functions `getwd()` and `setwd()` gets and sets the working directory. For this problem, set your working directory and run `getwd()` in an R command. Your TA will help you navigate your computer's file system.

Alternatively, and perhaps better, is to create an **R Project** and save all your class work as part of that project.

Problem 2: Importing data

Download the file `heights.csv` from the course website and place it into your working directory. Use `read.csv()` to read in the data from the file into an R object:

```
heights_df <- read.csv("heights.csv")
summary(heights_df)
```

```
##       id_.          gender               age           height
##  Min.   :  1.0   Length:506         Min.   :18.0   Min.   :143.0
##  1st Qu.:127.2   Class :character   1st Qu.:20.0   1st Qu.:163.0
##  Median :253.5   Mode  :character   Median :21.0   Median :171.0
##  Mean   :253.5                      Mean   :22.5   Mean   :170.8
##  3rd Qu.:379.8                      3rd Qu.:23.0   3rd Qu.:179.0
##  Max.   :506.0                      Max.   :61.0   Max.   :200.0
```

The object `heights_df` is a *data frame*, which we will talk much more about later. For now, know that `summary` summarizes the data in a data frame and that the *vector* called "height" can be accessed with the dollar sign '$':

```
heights_df$height
```

Find the sum of all the heights in this data frame.

Problem 3: String concatenation

1. Print `"hello world"` to the console.
2. Run the following code

```
x <- "hello"
y <- "world"
```

In some languages, adding two strings will concatenate them. In other words, `x + y` would return `"helloworld"`. Does this work in R? Explore the functions `paste` and `paste0` to see how to concatenate strings in R.

Problem 4: Vector coercion

You might know from linear algebra that a *vector* represents change in the form of a magnitude and a direction, often visualized with an arrow drawn in space. While this idea is useful in R programming as well, in this class a vector is simply a sequence of values.

Atomic vectors are vectors in which all values are the same data type. The other type of vector is the *list*, which we may talk about later. Lists can hold elements of different data types.

Create an atomic vector of the elements 1 through 10.

```
x <- 1:10
```

Now, change the fifth element of `x` to be the string `"cat"`. What happens to the other elements of `x`?

This is called *coercion*. Try this with logical data types (`TRUE`, `FALSE`) as well and establish the coercion hierarchy (or just look it up on StackOverflow).