Assignment 3 STATS 220 Data Technologies Due 5:00pm Friday 12 June

All questions in this assignment relate to bus status information that was collected from Auckland Transport[1]

The data are available from Canvas as a single ZIP file (`auckland-transport.zip`).

Each file represents a snapshot of bus status information, where each snapshot was taken approximately 10 minutes apart. Within each file, each row of data gives the status of one bus. The `delay` is how many seconds ahead or behind schedule the bus is (a positive value is behind schedule and a negative value is ahead of schedule). The `stop.id` identifies which bus stop the delay applies to. The `stop.sequence` is the order of the bus stop within the bus route. The `route` identifies the bus route. The first few lines of the file collected at 22:04 on March 9 2017 are shown in Figure 1.

There are **8** questions in this assignment, each requiring you to write some R code to work with this data set. All questions are worth equal marks.

```
"delay","stop.id","stop.sequence","route"
-271,"8703",32,"31201-20170214155712_v51.8"
-104,"8442",28,"26702-20170214155712_v51.8"
85,"4501",4,"98108-20170214155712_v51.8"
59,"7699",5,"30201-20170214155712_v51.8"
240,"8316",21,"24902-20170214155712_v51.8"
162,"8513",27,"27701-20170214155712_v51.8"

...
```

Figure 1: The first few lines of the bus data from 22:04 March 9 2017.

NOTE: You should submit a file (`assignment3.R`) containing R code that assigns the appropriate values to the appropriately named symbols. I will mark your code by running your code and inspecting the values that have been assigned to the relevant symbols.

NOTE: You should submit your answers online via Canvas.

[1]The data were obtained by Thomas Lumley using the Auckland Transport Web API, `https://api.at.govt.nz/`, and are made available for personal use only (see `https://api.at.govt.nz/terms/`).

Each question has an indication of when we will have covered the relevant material in the course. For example, "(Week 8)" means that you should be able to attempt the question by the end of Week 8 of this semester.

NOTE: You shall first create a new RStudio project named `stats220-assignment3` for this assignment. You shall place all the csv files included in the `auckland-transport.zip` under the `stats220-assignment3` folder, without creating any subfolders.

1. (Week 7)
   Use the `read.csv()` function to read the file `"2017-03-09-22-04.csv"` into R. Your code should assume that the file is in the current project directory.

   You should end up with a **data frame** called `bus1` that looks like this:

```
> head(bus1)

  delay stop.id stop.sequence                      route
1  -271    8703            32 31201-20170214155712_v51.8
2  -104    8442            28 26702-20170214155712_v51.8
3    85    4501             4 98108-20170214155712_v51.8
4    59    7699             5 30201-20170214155712_v51.8
5   240    8316            21 24902-20170214155712_v51.8
6   162    8513            27 27701-20170214155712_v51.8

> dim(bus1)

[1] 158   4
```

2. (Week 8)
   Use the `min()` and `max()` functions to calculate the largest ahead-of-schedule value and the largest behind-schedule value, *in minutes*, in the data frame `bus1` and assign the results to the symbols `aheadMax` and `behindMax`.

   You should end up with two **numeric vectors** called `aheadMax` and `behindMax` that look like this:

```
> aheadMax

[1] -12.21667

> behindMax

[1] 50
```

Hints:

If a data frame named `df` has a column named `x`, we can extract just the column `x` from the data frame with the expression `df$x`.

3. (Week 9)
Generate a set of file names for all of the files in the bus data set. You can assume that the files are all in the current project directory *and* that they are the only files in the current project directory that have names ending in `csv`.

You should end up with a **character vector** called `filenames` that looks like this:

```
> head(filenames)

[1] "2017-03-09-05-14.csv" "2017-03-09-05-24.csv" "2017-03-09-05-34.csv"
[4] "2017-03-09-05-44.csv" "2017-03-09-05-54.csv" "2017-03-09-06-04.csv"

> length(filenames)

[1] 101
```

4. (Week 9)
Write a loop that reads in all of the CSV files, calculates the largest ahead-of-schedule value and the largest behind-schedule value, *in minutes*, from each file, and determines overall largest values across all of the files. Assign the results to the symbols `amax` and `bmax`. Your code should assume that the files are in the current project directory.

You should end up with two **numeric vectors** called `amax` and `bmax` that look like this:

```
> amax

[1] -30

> bmax

[1] 50
```

Hints:

Some intermediate steps on the way to the final answer for this quesion might include ...

... a loop that prints out all file names.

... a loop that prints out the largest values for each of the files.

An example of a loop that updates a value each time the loop runs is shown below:

```
# Initialize overall maximum to small value
maximum <- -Inf
for (i in 1:5) {
  # Generate random number
  newValue <- rnorm(1)
  print(newValue)
  # If random number larger than current maximum, update maximum
  if (newValue > maximum) {
    maximum <- newValue
  }
}
print(maximum)
```

The largest ahead-of-schedule value and the largest behind-schedule value are both suspiciously whole numbers of minutes. This suggests that the AT monitoring system may have a cut-off for how long a bus can be ahead of or behind schedule. If a bus is 50 minutes late, will it ever arrive?

We will now look at excluding those suspicious delay values from our analysis.

5. (Week 10)
   Working just with the `bus1` data frame, use arithmetic and comparison operators to calculate which delay values are exact multiples of 60 seconds.

   You should end up with a **logical vector** called `multipleDelay` that looks like this:

```
> multipleDelay

  [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
 [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
[109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[121]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[133] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
[145] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[157] FALSE FALSE
```

Hints:

The `%%` operator in R gives the remainder on integer division. This value is zero when the numerator is an exact multiple of the divisor.

```
> 1:5 %% 3

[1] 1 2 0 1 2
```

6. (Week 11)
   Working just with the `bus1` data frame, use subsetting (amongst other things) to calculate the largest ahead-of-schedule value and the largest behind-schedule value, *in minutes*, for the `bus1` data frame, *excluding* delays that are exact multiples of 60 seconds.

   You should end up with two **numeric vectors** called `aheadMaxNotMult` and `behindMaxNotMult` that look like this:

   ```
   > aheadMaxNotMult

   [1] -12.21667

   > behindMaxNotMult

   [1] 25.55
   ```

7. (Week 11)
   Write a loop that reads in all of the CSV files and calculates the largest ahead-of-schedule value and the largest behind-schedule value, *in minutes*, across all files, *excluding* delays that are exact multiples of 60 seconds.

   You should end up with two **numeric vectors** called `amaxNotMult` and `bmaxNotMult` that look like this:

   ```
   > amaxNotMult

   [1] -27.93333

   > bmaxNotMult

   [1] 49.36667
   ```

8. (Week 11)
   Write a loop that reads in all of the CSV files and calculates the largest ahead-of-schedule value and the largest behind-schedule value, *in minutes*, across all files, *excluding* delays that are exact multiples of 60 seconds, *only for* delays at the first stop on a route.

   You should end up with two **numeric vectors** called `amaxStop1` and `bmaxStop1` that look like this:n

   ```
   > amaxStop1

   [1] -19.68333

   > bmaxStop1

   [1] 49.36667
   ```

- **[EXTRA for EXPERTS - NO MARKS]**

  From Question 5 onward, we excluded *all* delays that were an exact multiple of 60 seconds. This approach would not be ideal if we were looking for something other than the maximum or minimum delays (e.g., if we were calculating the average delay within a file). It is perfectly plausible for a bus to be exactly 5 minutes late.

  Write code that will only exclude a delay if it is the largest or smallest delay *and* it is an exact multiple of 60 seconds. Note that, if we exclude a delay, we must also check the new maximum value once that delay has been removed (and so on).

  Working just with the `bus1` data frame, the largest ahead-of-schedule value and the largest behind-schedule value (in minutes) will be the same as from Question 6 ...

```
[1] -12.21667
```

```
[1] 25.55
```

  ... *but* the average delay using the data from Question 6 is this ...

```
[1] 0.04660297
```

  ... while the correct average delay for this question is this ...

```
[1] -0.01765351
```