UNIVERSITY OF THE PHILIPPINES VISAYAS
COLLEGE OF ARTS AND SCIENCES
DIVISION OF PHYSICAL SCIENCES AND MATHEMATICS

CMSC 197 (Introduction to Data Science)
1st Semester AY 2021-2022

FIRST MINI-PROJECT

ACADEMIC INTEGRITY

As a student of the University of the Philippines, I pledge to act ethically and uphold the value of honor and excellence. I understand that suspected misconduct on given assignments/examinations will be reported to the appropriate office and if established, will result in disciplinary action in accordance with University rules, policies and procedures. I may work with others only to the extent allowed by the Instructor.

INSTRUCTIONS:

1. Using RStudio, create an R file with filename **FirstMiniProj.R** . Include all the codes for all the problems in this mini-project in that file. Make sure to label which part of your code is intended for each item. Include a sample code on how to run each item.

2. Commit your completed R file into YOUR git repository and push your git branch to the GitHub repository under your account.

3. In a blank document, include the URL to your GitHub repository that contains the completed R code for the project. In addition, include also the 40-character SHA-1 hash (as string of numbers from 0-9 and letters from a-f) that identifies the repository commit that contains the version of the files you want to submit. You can do this in GitHub by doing the following:

a. go to your GitHub repository web page

b. click on the "?? commits" link where ?? is the number of commits you have in the repository. For example, if you made a total of 10 to this repository, the link should say "10 commits"

c. you will see a list of commits that you have made to your repository. The most recent commit is at the very top. If this represents the version of the file you want to submit, then just click the "copy to clipboard" button on the right hand side that should appear when you hover the SHA-1 hash. Paste this SHA-1 hash into the blank document that you will submit.

4. Save as **LastName_FirstLetterOfFirstName_FirstMiniProj.doc** or

LastName_FirstLetterOfFirstName_FirstMiniProj.pdf.

5. Submit your document file in the LMS **on** or **before October 25 11:59 PM.**

PROBLEMS:

1. Download the zipped file **rprog_data_specdata.zip** from the LMS. Unzip the file and create a directory 'specdata' in your local file. DO NOT make any modifications in the files included in the zipped file. The zip file contains 332 comma-separated value (CSV) containing pollution monitoring data for fine particulate matter (PM) air pollution at 332 location in the US. Each file contains data from a single monitor and the ID number for each monitor is contained in the file name. For example, data for monitor 332 is contained in the file '332.csv'. Each file contains 3 variables:

- `Date:` the date of the observation in YYYY-MM-DD format (year-month-day)
- `sulfate:` the level of sulfate PM in the air on that date (measured in microogranisms per cubic meter)
- `nitrate:` the level of nitrate PM in the air on that date (measured in microorganisms per cubic meter)

In each file, there are many days where either sulfate or nitrate (or both) are missing (NA). This is common with air pollution monitoring data in the US.

Write a function named **pollutantMean** that calculates the mean of a pollutant (sulfate or nitrate) across a specified list of monitors. The function **pollutantMean** takes 3 argument: **directory, pollutant,** and **id**. Given a vector monitor ID numbers, **pollutantMean** reads that monitor's particulate matter data from the directory specified in the **directory** argument and returns the means of the pollutant across all of the monitors, ignoring any missing values coded as NA. **Note**: Include comments explaining how per line works. A prototype of the function is as follows:

```
pollutantmean <- function(directory, pollutant, id = 1:332) {
  ## 'directory' is a character vector of lenth 1 indicating
  ## the location of the CSV files

  ## 'pollutant' is a character vector of length 1 indicating
  ## the name of the pollutant for which we will calculate the
  ## mean; either 'sulfate' or 'nitrate'

  ## 'id' is an integer vector indicating the monitor ID numbers
  ## to be used

  ## Return the mean of the pollutant accross all monitors list
  ## in the 'id' vector (ignoring NA values)
  ## You do not need to round the results.
}
```

Example output:

```
pollutantmean("specdata", "sulfate", 1:10)
```

```
## [1] 4.064128
```

```
pollutantmean("specdata", "nitrate", 70:72)
```

```
## [1] 1.732979
```

```
pollutantmean("specdata", "nitrate", 23)
```

```
## [1] 1.280833
```

2. Modify your code from item number 1, this time, write a function named **complete** that reads a directory full of files and reports the number of completely observed cases in each data file. The function should return a data frame where the first column is the name of the file, and the second column is the number of complete cases. A prototype of this function follows:

```
pollutantmean <- function(directory, id = 1:332) {
  ## 'directory' is a character vector of lenth 1 indicating
  ## the location of the CSV files

  ## 'id' is an integer vector indicating the monitor ID numbers
  ## to be used

  ## Return a data frame of the form:
  ## id    nobs
  ## 1     117
  ## 2     1041
  ## ...
  ## where id is the monitor ID number and nobs is the number
  ## of complete cases
}
```

Example output:

```
complete("specdata", 1)
```

```
##   id nobs
## 1  1  117
```

```
complete("specdata", c(2, 4, 8, 10, 12))
```

```
##   id nobs
## 1  2 1041
## 2  4  474
## 3  8  192
## 4 10  148
## 5 12   96
```

```
complete("specdata", 30:25)
```

```
##   id nobs
## 1 30  932
## 2 29  711
## 3 28  475
## 4 27  338
## 5 26  586
## 6 25  463
```

```
complete("specdata", 3)
```

```
##   id nobs
## 1  3  243
```

3. By modifying your code from item 1 and 2, write a function named **corr** that takes a directory of data files and a threshold for complete cases and calculates the correlation between sulfate and nitrate for monitor locations where the number of completely observed cases (on all variables) is greater than the threshold. The function should return a vector of **correlations** for the monitors that meet the threshold requirement. If no monitors meet the threshold requirement, then the function should return a numeric vector of length 0. A prototype of this function follows

```
corr <- function(directory, threshold = 0) {
  ## 'directory' is a character vector of lenth 1 indicating
  ## the location of the CSV files

  ## 'threshold' is a numeric vector of length 1 indicating the
  ## number of completely observed observations (on all variables)
  ## required to compue the correlation between
  ## nitrate and sulfate; teh default is 0

  ## Return a numeric vecor of correlations
  ## Do not round the result.
}
```

For this function, you will need to use the **`cor()`** function in R which calculates the correlation between two vectors. Please read the help page for this function via **`?cor`** and make sure that you know how to use it.

Example output: *Note: your output might differ slightly from the example below*

```
cr <- corr("specdata", 150)
head(cr); summary(cr)
```

```
## [1] -0.01895754 -0.14051254 -0.04389737 -0.06815956 -0.12350667 -0.07588814
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.21060 -0.04999  0.09463  0.12530  0.26840  0.76310
```

```
cr <- corr("specdata", 400)
head(cr); summary(cr)
```

```
## [1] -0.01895754 -0.04389737 -0.06815956 -0.07588814  0.76312884 -0.15782860
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.17620 -0.03109  0.10020  0.13970  0.26850  0.76310
```

```
cr <- corr("specdata", 5000)
head(cr); summary(cr) ; length(cr)
```

```
## NULL
```

```
## Length  Class   Mode
##      0   NULL   NULL
```

```
## [1] 0
```

```
cr <- corr("specdata") # default threshold value  is ZERO
head(cr); summary(cr) ; length(cr)
```

```
## [1] -0.22255256 -0.01895754 -0.14051254 -0.04389737 -0.06815956 -0.12350667
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.00000 -0.05282  0.10720  0.13680  0.27830  1.00000
```

```
## [1] 323
```

4. Download the zipped file **rprog_data_HospData.zip** from the LMS. Unzip the file in a directory that will serve as your working directory. When you start up R make sure to change your working directory to the directory where you unzipped the data,

The data come from the Hospital Compare website (http://hospitalcompare.hhs.gov) run by the U.S. Department of Health and Human Services. This dataset covers all major U.S. hospitals. The website contains a lot of data and we will only look at a small subset. The zip file contains 3 files:

- `outcome-of-care-measures.csv`: contains information about 30-day mortality and readmission rates for heart attacks, heart failure, and pneumonia for over 4,000 hospitals.
- `hospital-data.csv`: contains information about each hospital
- `Hospital_Revised_Flatfiles.pdf`: descriptions of the variables in each file (i.e the code book).

A description of the variables in each of the files is in the included PDF file named `Hospital_Revised_Flatfiles.pdf`. This document contains information about many other files that are not included for the project. You will want to focus on the variables for Number 19 ("Outcome of Care Measures.csv") and Number 11 ("Hospital_Data.csv"). In particular, the numbers of the variables for each table indicate column indices in each table (i.e. "Hospital Name" is column 2 in the `outcome-of-care-measures.csv` file).

Read the outcome data into R via the **`read.csv`** function and look at the first few rows.

```
outcome <- read.csv('outcome-of-care-measures.csv', colClasses = "character")
head(outcome)
```

There are many columns in this dataset. You can see how many by typing **`ncol(outcome)`** and you can see the number of rows with the **nrow** function. In addition, you can see the names of each column by typing **`names(outcome`** , which will show you the names which are included in the PDF document.

To make simple histogram of the 30-day death rates from heart attach (column 11 in the outcome dataset), run

```
outcome[, 11] <- as.numeric(outcome[, 11])
## You may get a warning about NAs being introduced; that is okay
hist(outcome[, 11])
```

Because we originally read the data in as character (by specifying colClasses = "character"), we need to coerce the column to be numeric. You may get a warning about NAs being introduced but that is okay.

Given the code above, modify it so that you can plot the 30-day mortality rates for heart attack given the dataset `outcome-of-care-measures.csv`.

Your output should look like this: