Unitec Institute of Technology
TE WHARE WĀNANGA O WAIRAKA

School of Computing & Information Technology

ISCG8026 Introduction to Data Science

Semester 1 – 2020

R Programming Assignment A
Due Date: 23 April 2020, 8:30am
Weight: 30%

Introduction

This is Part A of the ISCG 8026 course assignment and is worth 30% of your final grade. For this programming assignment, you will write five functions, four of which are meant to interact with dataset that accompanies this assignment. The dataset is contained in a zip file **rprog_assignment_dataset.zip**.

Data

The zip file containing the data can be downloaded from Moodle.

The zip file has 332 comma-separated-value (CSV) files containing pollution monitoring data for fine particulate matter (PM) air pollution at 332 locations. 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 200 is contained in the file "200.csv". Each file contains three 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 micrograms per cubic meter)
- nitrate: the level of nitrate PM in the air on that date (measured in micrograms per cubic meter)

You will need to unzip this file and create the directory 'specdata'. Once you have unzipped the zip file, do not make any modifications to the files in the 'specdata' directory. You need the dataset for Part 1-4. In each file you'll notice that there are many days where either sulfate or nitrate (or both) are missing (coded as NA). This is common with air pollution monitoring data.

Part 1

[6 Marks]

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 three arguments: 'directory', 'pollutant', and 'id'. Given a vector monitor ID numbers, 'pollutantmean' reads that monitors' particulate matter data from the directory specified in the 'directory' argument and returns the mean of the pollutant across all of the monitors, ignoring any missing values coded as NA. A prototype of the function is as follows:

```
pollutantmean <- function(directory, pollutant, id = 1:332) {
        ## 'directory' is a character vector of length 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 across all monitors list
        ## in the 'id' vector (ignoring NA values)
        ## NOTE: Do not round the result!
}
```

Please save your code to a file named pollutantmean.R. Examples of calling the function:

```
source("pollutantmean.R")

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

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

Part 2

[6 Marks]

Write a function 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

```
complete <- function(directory, id = 1:332) {
        ## 'directory' is a character vector of length 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
}
```

Please save your code to a file named complete.R. To run the submit script for this part, make sure your working directory has the file complete.R in it. Examples of calling the function:

```
source("complete.R")

complete("specdata", 1)

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

Part 3

[6 Marks]

Write a function 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 length 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 compute the correlation between
        ## nitrate and sulfate; the default is 0

        ## Return a numeric vector of correlations
        ## NOTE: 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.

Please save your code to a file named corr.R. To run the submit script for this part, make sure your working directory has the file corr.R in it. Examples of calling the function:

```
source("corr.R")
source("complete.R")

cr <- corr("specdata", 150)
```

Part 4

[6 Marks]

Write a function named 'pollutantvector' that returns a vector of those pollutants (sulfate or nitrate) whose values are greater than 'p', across a specified list of monitors. The function 'pollutantvector' takes four arguments: 'directory', 'pollutant', 'id' and 'p'. Given a vector monitor ID numbers, 'pollutantvector' reads that monitors' particulate matter data from the directory specified in the 'directory' argument and returns the ones more than a certain value ('p') across all of the monitors, ignoring any missing values coded as NA.

Please save your code to a file named pollutantvector.R. Examples of calling the function:

```
source("pollutantvector.R")

pollutantvector("specdata", "sulfate", 1:35, 0.5)
```

Part 5

[6 Marks]

Write a function that prompts a user to choose an operation between six available operations: 1) Add, 2) Subtract, 3) Multiply, 4) Divide, 5) Factors and 6) Prime number. The first four operations will ask user to provide two numbers and add, subtract, multiply and divide them accordingly. The fifth operation calculates the factors of a number and sixth operation checks if a number is prime.

Please save your code to a file named calculator.R. The output should look something like this when the user runs the function:

```
[1] "******Simple R Calculator - Select operation: ******"
[1] "1.Add"
[1] "2.Subtract"
[1] "3.Multiply"
[1] "4.Divide"
[1] "5.Factors"
[1] "6.Prime"


Enter choice [1/2/3/4/5/6]: 4   #prompting the user to select an operation
Enter first number: 20          #prompting the user to enter the first number
Enter second number: 4 #prompting the user to enter the second number [1]
"20 / 4 = 5"



[1] "******Simple R Calculator - Select operation: ******"
[1] "1.Add"
[1] "2.Subtract"
[1] "3.Multiply"
[1] "4.Divide"
[1] "5.Factors"
[1] "6.Prime"


Enter choice [1/2/3/4/5/6]: 5   #prompting the user to select an operation
Enter the number: 120           #prompting the user to enter the input


[1] "The factors of 120 are:"
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 8
[1] 10
[1] 12
[1] 15
[1] 20
[1] 24
[1] 30
[1] 40
[1] 60
[1] 120
```

For this function you will need to use the '**readline**' function in R to take input from the user (terminal). Please read the help page for this function via '?readline' and make sure that you know how to use it.

Late Submission of Assignments

Assignments submitted after the due date and time without having received an extension through Special Assessment Circumstances (SAC) will be penalised according to the following:

- 10% of marks deducted if submitted within 24hrs of the deadline
- 20% of marks deducted if submitted after 24hrs and up to 48hrs of the deadline
- No grade will be awarded for an assignment that is submitted later than 48hrs after the deadline

Assignments submitted in more than 48 hours late will not be marked unless Special Assessment Circumstances apply. So, it is better to submit an incomplete assignment on time.

Special Assessment Circumstances

A student, who due to circumstances beyond his or her control, misses a test, final exam or an assignment deadline or considers his or her performance in a test, final exam or an assignment to have been adversely affected, should complete the Special Assessment Circumstances (SAC) form available from Student Central.
Within any semester, a student may have only one SAC per course.
When requesting an SAC for an assignment, the SAC application form must be submitted (along with the work completed to date) within the time frame of the extension requested; i.e. if the Doctor's certificate is for one (1) day, then the SAC application form and work completed must be submitted within one (1) day.

Assistance to other Students

Students themselves can be an excellent resource to assist the learning of fellow students, but there are issues that arise in assessments that relate to the type and amount of assistance given by students to other students. It is important to recognise what types of assistance are beneficial to another's learning and also what types of assistance are unacceptable in an assessment.

Beneficial Assistance

- Study Groups.
- Discussion.
- Sharing reading material.
- Testing another student's programming work using the executable code and giving them the results of that testing.

Unacceptable Assistance

- Working together on one copy of the assessment and submitting it as own work.
- Giving another student your work.
- Copying someone else's work. This includes work done by someone not on the course.
- Changing or correcting another student's work.
- Copying from books, Internet etc. and submitting it as own work. Anything taken directly from another source must be acknowledged correctly: show the source alongside the quotation.

Do you want to do the best that you can do on this assignment and improve your grades? You could:
• Talk it over with your lecturer
• Visit Student Success and Achievement for learning advice and support (in Te Puna)
• Visit the Centre for Pacific Development and Support