Western Sydney University

Programming for Data Science (COMP7024) Assignment

2022 Q1

Due Sunday 27th of February 2022

Introduction

This assignment consists of four questions, each of equal value, giving a total contribution of 40% to this subject. The beginning of each question provides a breakdown of marks for each part in that question. For example, a breakdown of $(1 + 3 + 6 = 10)$ implies a question consisting of three parts, where the first, second and third parts are worth 1, 3 and 6 marks respectively.

IMPORTANT

R and only packages (that is, ***R*** libraries) described in the lectures and tutorials for this subject can be used for generating answers for this assignment! In addition, ***R***Markdown must not be used[1]. Penalties may apply for non-conformance.

Answer structure

In doing this assignment, you should not need to use the maximum word limit declared in the Learning Guide. Note that marks are *not* awarded for using many words, rather, for using an economy of words and only stating what is relevant to the question being answered. Consider the old adage – "less can be more". Therefore, show you know what is relevant and have mastered the ability to get to the point using few, simple words and clear sentences. Seek to also apply this philosophy to the code you write.

Your answers to this assignment are to be provided in a *single* ***R*** script file. All material in your script file should be logically organised, so that related material can be easily and quickly located. Clearly identify yourself in this file, as a minimum: full name and student ID, as comments at the beginning of the file.

R script file

Textual answers should be included as comments in your script file, refer to listing 1 for examples on including comments. The comments in your script file should be:

- Brief and to the point
- Stating a high level perspective
- Stating what is not immediately obvious, but worth mentioning

[1] In short, only use ***R*** and ***R*** packages described in the lectures and tutorials for this subject. You are also not allowed to use ***R***Markdown for this assignment. All these requirements will also apply for the exam.

```r
# In an R script file, comments are prefixed with the hash symbol
# A line with just a comment on it

# Generate a distribution of mean values from a sequence of digits
d <- replicate(1000,
  {
    s <- sample(0:9, replace = TRUE)  # Generate a sequence of 10 numeric digits
    mean(s)   # A comment to end a line with R code on it
  })

hist(d, main = 'Distribution of means')   # Show distribution of means

# Of cause you would use smarter comments than those used here
# Only state what is not immediately obvious
```

Listing 1: Some ***R*** code with comments (shown in green)

Be brief and to-the-point with respect to comments. The approach described in this section should be the same as used for the exam. **Make judicious use of comments a priority in doing this assignment.** After all, comments are meant to communicate important and useful details. Make sure you also communicate well through wise choices in variable and function names. Also make wise decisions regarding the layout of *everything* inside your ***R*** script file. Note if things go wrong, good organisation and comments can help you, since they can show if appropriate logic was intended.

Plagiarism

This is an individual effort assignment, therefore the answers you provide *must* be your own. You may learn from others, but the *understanding* claimed by your assignment must be *yours*. If you include any material in this assignment that is not your own, you *must* acknowledge that fact and declare the source of that material. **Be warned, your answers will be checked for plagiarism and if caught, significant penalties may apply.**

Submission

Once you have completed the assignment, you must upload your ***R*** script file via Turnitin; if you wish, you can also e-mail your ***R*** script file directly to me[2]. This maybe wise if you are having trouble with Turnitin or vUWS and are at risk of submitting late. Once you have e-mailed, seek to successfully submit via Turnitin. Be aware that you may need to rename your ***R*** script file by adding the extension ".txt", otherwise you may not be successful in submitting via Turnitin.

On a Windows machine you can easily add a ".txt" extension via file explorer. Select the "View" tab and tick "File name extensions", refer figure 1. Then select the file to be renamed, press F2 to enter edit mode and add ".txt" to the very end of the file name; do not remove the ".R" portion of the file name.

Hopefully a similar process is available on other platforms. Determine the method you will use and test it prior to submission.

You must submit your assignment no later than the due date declared on the first page of this assignment, otherwise late submission penalties will apply, as described in the section titled "Late submission penalties". Prior to the due date, you may replace a previously submitted version, but only the *last* submitted version will be marked!

[2] f.ubaudi@westernsydney.edu.au

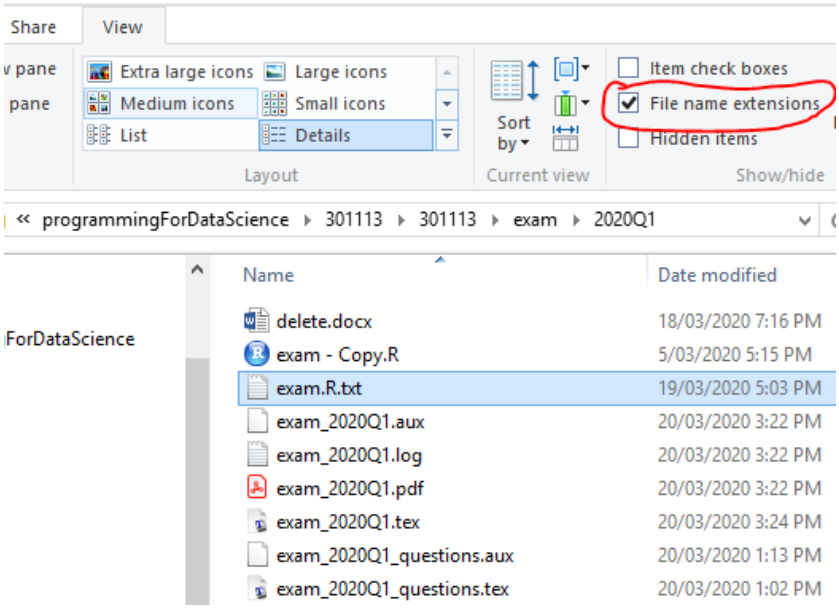

Figure 1: How to add ".txt" to the file extension on Windows

Late submission penalties

Late submission penalties exist. The contribution value of the assignment will reduce by 10% per day, for each day after the submission date; therefore four marks per day. For example, if your assignment is four days late, the maximum possible mark you can score for the assignment is 24 out of 40.

Question 1 (4 + 6 = 10)

Simple algorithms and functions

(i)

A Fibonacci Sequence of numbers is incredibly interesting and well studied, for reasons such as its ability to explain the "arrangement of leaves on a stem". A short sequence is

```
0, 1, 1, 2, 3, 5, 8, 13, 21, 34
```

Given a starting sequence of `(0, 1)`, every number is simply the sum of the previous two numbers. For example, the number 13, is the sum of 5 and 8. Write the following function to generate a Fibonacci sequence of length *len*.

```
generateFibSeq <- function(len = 10)
{
  # Generate a Fibonacci sequence of length len
  # This function returns a vector of length len

  # Missing code

  return(seq)
}
```

(ii)

A prime number is a natural number[3] greater than 1 that is not a product of two smaller natural numbers The first 25 prime numbers are

```
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71
73, 79, 83, 89, 97
```

Write the following function to generate a sequence of prime numbers of length *len*.

```
generatePrimeNumbers <- function(len = 10)
{
  # Generate a sequence of Prime numbers of length len
  # This function returns a vector of length len

  # Missing code

  return(seq)
}
```

[3]In mathematics, natural numbers are those used for counting, for example, there are 23 students in the class. Hence a natural number is simply a positive integer, which includes zero.

Question 2 (4 + 4 + 2 = 10)

Simulation exercise

(i)

Using the following equation (or Equation 1)

$$Area = \pi \times r^2 \tag{1}$$

we can calculate the exact $Area$ of a circle, only knowing its radius r. For this exercise, we will pretend to not know about Equation 1 and therefore seek to accurately estimate the area of a circle through the use of simulation.

Add the necessary code and appropriate comments, to the following function prototype, in order to estimate the area of a circle with accuracy proportional to the number of trials ($nTrials$) used.

```
estCircleArea <- function(r, nTrials = 10^5)
{
  # Estimate the area of a circle with radius r,
  # using nTrials or repetitions.

  # Missing code

  return(area)
}
```

Make use of Pythagoras's Theorem, shown in Equation 2, in order to complete this task.

$$r^2 = x^2 + y^2 \tag{2}$$

(ii)

Consider a pair of dice that are rolled in some game. However, imagine the dice are loaded (or biased) in the following way:

- For one die, the result of a roll is proportional to its value. Specifically, getting a 2 is twice as likely as getting a 1 and getting a 3, is three times more likely than getting a 1, etc.
- For the other die, every side, except a 6, is equally likely to occur, while a 6 is twice as likely to occur as any other face of that die.

Use simulation to estimate the probabilities of each of the 36 possible outcomes for rolling that pair of dice.

(iii)

Present your findings from sub-question(ii) in two ways:

1. Generate a simple, but informative table and display its contents vertically, where the order of the rows is according to the probability of getting the result of the roll shown in that row
2. Use what you consider to be the most appropriate visualisation and very briefly state why you chose it

Question 3 (4 + 6 = 10)

Data cleaning and repair

This question revolves around the task of identifying bad data and as an additional task, seeking to repair that data in some simple and reasonable way. The dataset for this task, located in $vUWS$, is contained in the file called "irisBad.csv", which is a form of the *iris* dataset provided within ***R***. This dataset has the same dimensionality as *iris*, but some numeric values are either missing, hence containing *NA*; while some values within the dataset are outliers.

In performing this task, you are encouraged to produce code that is well organised, documented (through judicious use of comments), modular and reusable. For example, if you need to perform the same task in multiple places in your program, that task should exist within a function you create. An example of the virtues of re-usability is the task of saving data in a particular format to a disk file. This task should be performed by a function, that way the data can be saved exactly the same way, every time, from multiple places in your program.

In sub-question(i), you will be asked to locate and remove bad data, while sub-question(ii) is about adding the ability to repair data you removed. Therefore, it would be wise to try and organise your solution to sub-question(i), so that as much of the code as possible, created for sub-question(i), is reused in sub-question(ii).

The task of identifying outliers has been touched upon in the tutorial of week 2. Fundamentally you simply need to determine some reasonable limits, or window, within which you consider good data to be located. However, since the *iris* dataset consists of sub-groups, the three species of iris flowers, you need to determine such limits for each species. Data repair must also consider species. For example, determine some simple but reasonable way to determine a replacement value per species, to replace either an outlier or an *NA*. This task is actually quite straight forward, although potentially tedious, hence the need for organised code.

(i)

Your first task is simply to identify any row in the dataset that has at least one problem: either missing data (an *NA*), or values that are considered outliers. Any such row should be removed from your dataset[4]. Once you have cleaned the data, save the data into a file called "irisBad_cleaned.csv".

(ii)

This task builds on what you did in sub-question(i), here you add some simple form of data repair, so that none of the rows in the dataset "irisBad.csv" are lost. Once you have repaired the data, save the data into a file called "irisBad_repaired.csv".

[4]In taking this approach, we assume that there is lots of data and we can therefore afford to lose data we do not trust. An additional motive for this is, that we consider not repairing data smarter, if we have plenty, rather than risking bad data repairs.

Question 4 (2 + 2 + 3 + 3 = 10)

Very simple classifier from scratch

Here you are going to get a taste of very simple machine learning by building your own classifier. *This is not about applying machine learning or statistics, it is about using* **R** *to build and test your very own and very simple classifier.* The classifier you build will be used to predict which species an iris flower belongs to, on the basis of a measurement. Hence you are only using the *iris* dataset, as provided within ***R***, for this task.

(i)

Produce a *single* visualisation consisting of four boxplots, where each boxplot shows, for a particular measurement, how the measurements vary for each species. So each boxplot corresponds to a single measurement, for example *Sepal.Length*. However every boxplot shows all three species of the iris flower.

(ii)

Using what you can see from the boxplots, pick a *single* measurement / flower variable and determine windows[5] (or fences, using Tukey's terminology), to predict a flower's species. For example, you might decide a flower is a

- *setosa* if $Petal.Length < x$ where $x > 0$
- *virginica* if $Petal.Length > y$ where $y > 0$
- is otherwise a *versicolor*

Create a function called $predictSpecies(x)$ to predict a flower's species.

(iii)

Write a function called $evaluatePrediction()$, which uses $predictSpecies(x)$ and determines, on the basis of the entire dataset, what proportion of predicts are correct. Very briefly comment on your findings?

(iv)

Create an upgraded version of $predictSpecies(x, y)$ so it uses two measurements. Does it result in better accuracy? Very briefly comment on your findings.

[5] You are not expected to find the most optimal window boundaries, just find half reasonable values.