STAT 2150 Assignment 1

Due Friday, May 17 at 11:59 PM

Thanh Dat Nguyen - THANHDAN

Instructions:

- Knit this file to pdf to see the questions in a more readable format.
- Do not change the code in the provided code chunks or in the questions. Simply fill in your answers in the code chunks that are missing code and answer the questions with text responses. Be sure when adding in text responses to never copy-paste symbols from outside of the document.
- Do not use any functions or methods that we have not learned in class.
- It is recommended that you knit to pdf after you fill in each code chunk.
- Your knit pdf file should show the result answering the question. To do this, after creating an R object, you should also print it in a new line within the code chunk, unless otherwise instructed.

Question 1:

Suppose a course has two quizzes, two term tests and a final exam. The weights of the course assessments are as follows:

- 15% for the two quizzes, each worth 7.5%
- 35% for the two tests, with the lower test score worth 15% and the higher test score worth 20%
- 50% for the final exam

Write a function that takes in a **vector** `x` of five percentage scores (numbers between 0 and 100) in the following format: (Quiz 1, Quiz 2, Test 1, Test 2, Final Exam). The function should return a list (a type of R object), where the first element of the list is the student's final percentage grade, the second element of the list is the student's letter grade (using letter grade cutoffs on our course syllabus), and the third element of the list is the higher test score.

Show that when you call your function with the following scores, it returns the expected output:

- If a student gets the following scores: (Quiz 1 = 85, Quiz 2 = 90, Test 1 = 80, Test 2 = 70, Final Exam = 85), the list should return 82.125 as the final percentage grade, A as the final letter grade, and 80 as the higher test score.
- If a student gets the following scores: (Quiz 1 = 65, Quiz 2 = 75, Test 1 = 60, Test 2 = 70, Final Exam = 55), the list should return 61 as the final percentage grade, C as the final letter grade, and 70 as the higher test score.

```
five_percentage_scores <- function(x){
  Quiz_1 <- x[1]
  Quiz_2 <- x[2]
  Test_1 <- x[3]
  Test_2 <- x[4]
  Final_Exam <- x[5]
  two_quizzes <- Quiz_1*0.075 + Quiz_2*0.075
```

```r
  if(Test_1 > Test_2){
    two_tests <- Test_1*0.2 + Test_2*0.15
    higher_score <- Test_1
  } else if(Test_1 < Test_2){
    two_tests <- Test_2*0.2 + Test_1*0.15
    higher_score <- Test_2
  } else {
    two_tests <- Test_1*0.175 # If both tests are equal, we can simply use one of them
    higher_score <- Test_1
  }

  final_exam <- Final_Exam*0.5

  final_grade <- sum(two_quizzes,two_tests,final_exam)

  if (final_grade > 90) {
    final_letter_grade <- "A+"
  } else if (final_grade >= 80 && final_grade < 90) {
    final_letter_grade <- "A"
  } else if (final_grade >= 75 && final_grade < 80) {
    final_letter_grade <- "B+"
  } else if (final_grade >= 70 && final_grade < 75) {
    final_letter_grade <- "B"
  } else if (final_grade >= 65 && final_grade < 70) {
    final_letter_grade <- "C+"
  } else if (final_grade >= 60 && final_grade < 65) {
    final_letter_grade <- "C"
  } else if (final_grade >= 50 && final_grade < 60) {
    final_letter_grade <- "D"
  } else {
    final_letter_grade <- "F"
  }

  list_grade <- list()
  list_grade[[1]] <- final_grade
  list_grade[[2]] <- final_letter_grade
  list_grade[[3]] <- higher_score

  return(list_grade)

}
student_1 <- c(85, 90, 80, 70, 85)
five_percentage_scores(student_1)
```

```
## [[1]]
## [1] 82.125
##
## [[2]]
## [1] "A"
##
## [[3]]
## [1] 80
```

Question 2:

The `quakes` dataset is built-in to R. See documentation on the dataset by typing `?quakes` at the R console.

(a) Write R code that obtains the average number of stations reporting on earthquakes near Fiji for earthquakes that have a magnitude at least 5.5.

```
?quakes
```

```
## starting httpd help server ... done
```

```
average_number <- quakes[which(quakes$mag >= 5.5),]
mean_mag <- mean(average_number$mag)
mean_mag
```

```
## [1] 5.665789
```

(b) Write R code that obtains the average number of stations reporting on earthquakes near Fiji for earthquakes that have a magnitude at least 5.5 and a depth of at least 600 kilometres.

```
average_number <- quakes[which(quakes$mag >= 5.5 & quakes$depth >= 600),]
mean_mag <- mean(average_number$mag)
mean_mag
```

```
## [1] 5.7
```

(c) Write the R code that creates a vector called `mag_cat` that re-codes the quantitative `mag` column of data into a categorical variable as follows:

- If the magnitude is greater than or equal to 5.5, code it as "High"
- If the magnitude is greater than or equal to 5.0 and less than 5.5, code it as "Medium"
- If the magnitude is less than 5.0, code it as "Low"

That is, each element of `mag_cat` should contain a value of "High", "Medium" or "Low", depending on the corresponding element of `mag` in the `quakes` data frame.

Finally, use the `table()` function of R to show how many earthquakes in the `quakes` data frame are classified as "High", "Medium" or "Low" magnitude. (The order of the output of the `table()` function does not matter.)

```
mag_cat <- rep(NA, nrow(quakes))  # Create an empty vector to store the categories

# Categorize earthquakes based on magnitude
mag_cat[quakes$mag >= 5.5] <- "High"
mag_cat[quakes$mag >= 5.0 & quakes$mag < 5.5] <- "Medium"
mag_cat[quakes$mag < 5.0] <- "Low"

# Count the occurrences of each category
magnitude_table <- table(mag_cat)
magnitude_table
```

```
## mag_cat
##   High    Low Medium
##     38    802    160
```

Question 3:

The Collatz conjecture is a mathematical hypothesis that states that every positive integer (1, 2, 3, ...) will eventually transform into 1 at the end of a sequence of simple arithmetic operations.

If the number we start with is odd, the next term in the sequence is 3 times the previous term plus 1. If the number we start with is even, the next term in the sequence is one half of the previous term. At each step of the sequence, the same calculation is repeated: (a) if odd, multiply by 3 and add 1, (b) if even, divide by 2.

Sequences will be of varying length. For example, if we start with 6, since 6 is even, $6/2 = 3$, then 3 is odd, so 3*3+1 = 10, then $10/2 = 5$, then 5*3+1 = 16, then $16/2 = 8$, then $8/2 = 4$, then $4/2 = 2$, and $2/2 = 1$. We have arrived at 1, and the full sequence has 9 terms: (6, 3, 10, 5, 16, 8, 4, 2, 1).

If we start with 5, since 5 is odd, 3*5+1 = 16, then 16 is even, so $16/2 = 8$, then $8/2 = 4$, then $4/2 = 2$, and $2/2 = 1$. We have arrived at 1, and the full sequence has 6 terms: (5, 16, 8, 4, 2, 1).

Since we don't know how long the sequence will be, we can use a `while()` loop to continue the sequence until we arrive at 1.

For each of the first 20 integers (1, 2, ..., 19, 20) as starting points of the sequence, use the Collatz conjecture operations to create a vector `y` containing the number of terms in the 20 sequences. Use a `for()` loop to do the calculations for each of the 20 integers and use a `while()` loop inside the `for()` loop to continue the sequence until you arrive at 1. At the end, print the `y` vector so that its contents show up in your knit pdf file.

Some tips:

- Calculate the answer by hand for a few cases so you know what you expect your R code to do.
- If your answer only works for some cases, run your code chunk with the green arrow and look at the `y` vector in the environment to see when it failed to work.
- Use the R console to sequentially run your code line by line to see where it has stopped working, compared to what you can calculate by hand.
- If you see a red stop sign when you attempt to knit or use the green arrow, click on it. This means you have an infinite loop.
- Hint: I recommend using a line of code that says `last_term = next_term`. This will take the calculation of the "next term" and store it in the variable called `last_term` because the next term becomes the last term for the next step of the sequence.

```r
y <- double(20)
for (j in 1:20) {
  i <- j
  term <- 1

  while (i != 1) {
    if (i %% 2 == 0) {
      i <- i / 2
    } else {
      i <- i * 3 + 1
    }
    term <- term + 1
  }
  y[j] <- term
}

my_mat <- matrix(y, nrow = 20)  # Create a matrix with one row containing the values of y

print(my_mat)
```

```
##       [,1]
##  [1,]    1
##  [2,]    2
##  [3,]    8
##  [4,]    3
##  [5,]    6
##  [6,]    9
##  [7,]   17
##  [8,]    4
##  [9,]   20
## [10,]    7
## [11,]   15
## [12,]   10
## [13,]   10
## [14,]   18
## [15,]   18
## [16,]    5
## [17,]   13
## [18,]   21
## [19,]   21
## [20,]    8
```

Question 4:

The following code will generate a vector of 100 random values, where the smallest possible generated value is 0 and largest is about 9 (but the largest value you get may be only 6, 7 or 8, as each student will get a different vector of 100 values and if you re-compile this code, you will get a different vector each time).

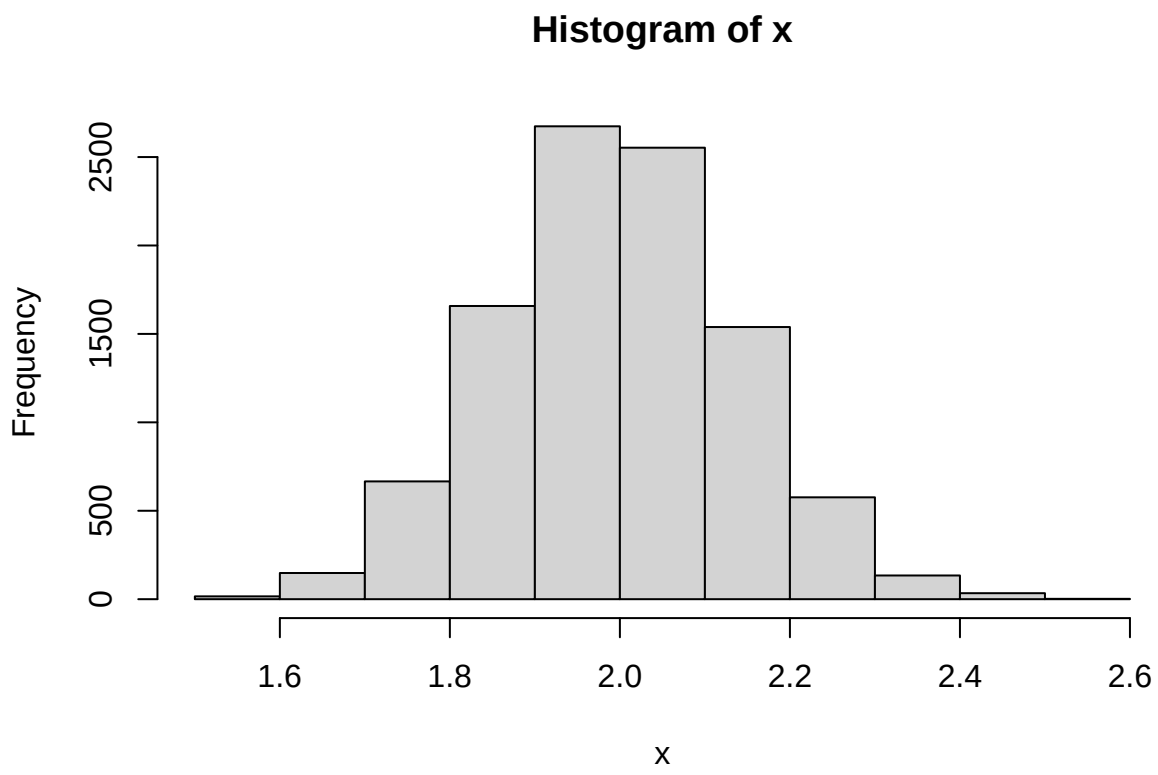
```
?rpois
rpois(100,2)
```

First, create a $10,000 \times 100$ matrix called `mymat` using a `for()` loop, where the `rpois(100,2)` line of code is used 10,000 times to populate each row of the matrix. (You are filling in a whole row of the matrix all at once with the above line of code.) Do not generate the data once, store it in an object, and then use that same data repeatedly 10,000 times. Instead, the data should be generated 10,000 times so that each vector of 100 values is different from each other.

Using a function from the `apply` family of functions (`apply`, `sapply` or `lapply`), create a vector `x` where each component of `x` is the mean of the corresponding row of `mymat`. Then make a histogram of the `x` vector. Do not print the matrix or the vector or else you will get very long output in your knit pdf file.

```
my_mat <- matrix(nrow=10000, ncol=100)
for(i in 1:10000){
    my_mat[i,] <- rpois(100, 2)
}

x <- apply(my_mat,1, mean)
hist(x)
```



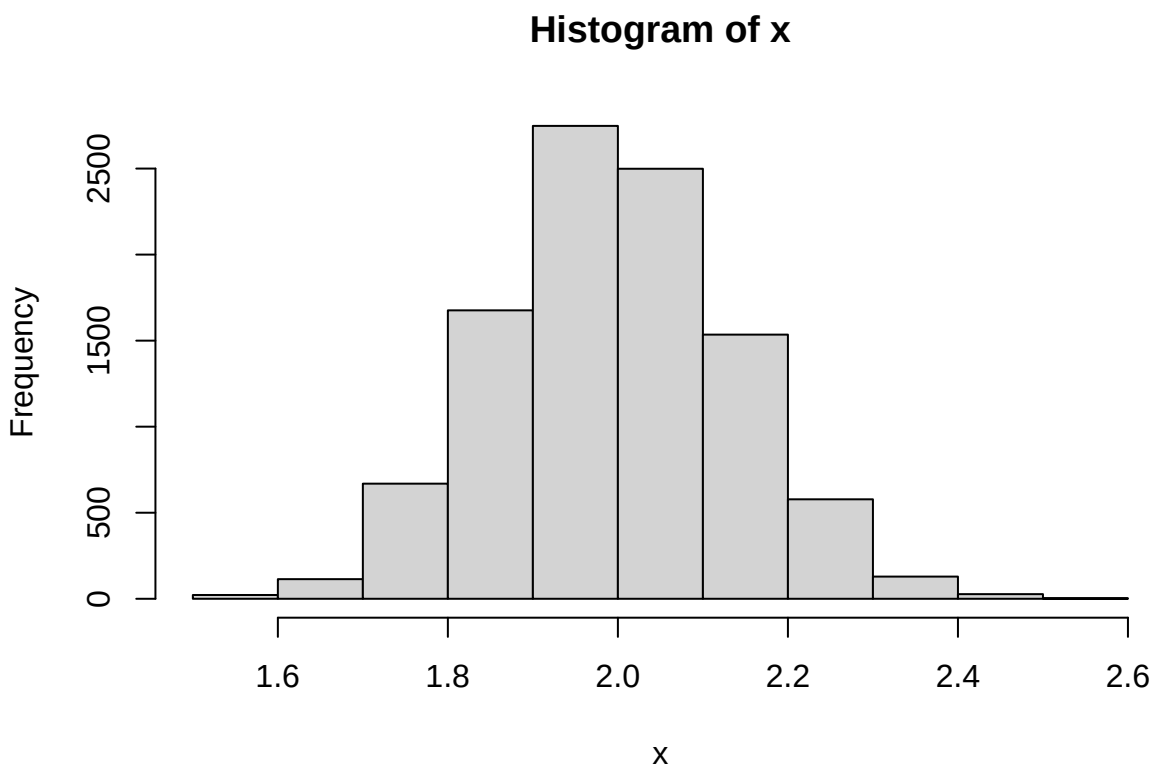
Now calculate what proportion of the values in the `x` vector are in the range [1.8,2.2).

```
prop_range <- length(x[x >= 1.8 & x < 2.2]) / length(x)
prop_range
```

```
## [1] 0.8404
```

Repeat the problem (creating the matrix, creating the vector `x`, making the histogram, and calculating what proportion of values in the `x` vector are in the range [1.8,2.2)), this time creating `x` **without** using a function from the `apply` family of functions. Do not print the matrix or the vector or else you will get very long output in your knit pdf file. Note that you will slightly different results when you repeat the problem as the two `x` vectors will be based on different randomly generated values.

```
my_mat <- matrix(nrow = 10000, ncol = 100)
for (i in 1:10000) {
  my_mat[i, ] <- rpois(100, 2)
}
x <- rowMeans(my_mat)
hist(x)
```




```
prop_range <- length(x[x >= 1.8 & x < 2.2]) / length(x)
prop_range
```

```
## [1] 0.8477
```