## 3. R/RStudio assessment (45pts – 45% of final mark)

This R assignment is split into 3 parts. The first part is about the general use of R/Rstudio, the second part about RNAseq and the third about ChIP-Seq. In these parts you will be asked to perform a number of tasks in R/RStudio and report them in your own markdown document.

Initial task: Create a new markdown document in *RStudio*, set the title to "Advanced Bioinformatics 2023 assessment", and insert an "author:" tag below the title, followed by your student id. Share your markdown document and html via your github account.

In the following, for each task, create a new heading called "Task X" for task X, and insert a new R code chunk that holds any code required. Make sure to evaluate the expression before saving to include the output in the html file. If you have multiple lines that produce outputs, you can split them into separate code chunks for increase clarity (but it is not necessary to pass the assessment). Please also explain your steps.

**General R/Rstudio assessment (33 pts)**

**3.1. Using the *sum*() function and : operator, write an expression in the code snippet to evaluate the sum of all integers between 5 and 55. (4pt)**

The following code is used to calculate the sum of all integers between 5 and 55

The first command is an easy way to list all integer between 5 and 55 in increments of 1 with ‘*a*’ being the variable. The sum functions allows us to calculate the sum of all integers between 5 and 55.

$ a <-seq(5,55,1)

$ sum(a)

Alternatively, a simpler way is following with a colon (:) operator dictating that sequence between 5 and 55 are to be collated

$ sum(5:55)

The answer is 1530.

Graphical user interface, text, application, email

Description automatically generated

**3.2. Write a function called *sumfun* with one input parameter, called *n*, that calculates the sum of all integers between 5 and *n*. Use the function to do the calculation for *n* = 10, *n* = 20, and *n* = 100 and present the results. (4pt)**

The following *sumfun* function is generated using input parameter of n, where *sumfun* defines the function.

*n* is the input parameter

The operation is enclosed within the curly brackets.

$ sumfun <- function(n) {sum(5:n)}

The output of n =10 is 45 using the following commands

$ sumfun <- function(n) {sum(5:n)}

$ sumfun(10)

The output of n=20 is 200 using the following code

$ sumfun <- function(n) {sum(5:n)}

$ sumfun (20)

The output of n=100 is 5040 using the following code

$ sumfun <- function(n) {sum(5:n)}

$ sumfun (100)

The following figure is a snapshot of the image displaying the codes being used.

Graphical user interface, text, application

Description automatically generated

**3.3. The famous Fibonacci series is calculated as the sum of the two preceding members of the sequence, where the first two steps in the sequence are 1, 1. Write an R script using a for loop to calculate and print out the first 12 entries of the Fibonacci series. (4pt)**

The for loop construction uses the following format

For (*name* in *expression1*) *expression2*

Where *name* is the loop variable which we have assumed to be ‘i’ in this case, *expression1* is the vector expression or sequence (3:n, in our instance, where n is the number of entries needed to be printed, i.e.,12) and *expression2* is a grouped expression with sub-expressions written in terms of the loop variable’i’ in our instance.

*name*: i

*expression1*: 3:n

*expression2*: fib[i] <-fib[i-1]+fib[i-2]

In order to calculate the first 12 entries, we set the n variable to 12; then, we define the values of the first two steps in the sequence as 1and 1. Then using the for loop command we used the aforementioned loop variables and vector expression/sequence to evaluate the statement.

$ n <- 12

$ fib <- numeric(n)

$ fib[1] <- 1

$ fib[2] <- 1

$ for (i in 3:n){fib[i] <-fib[i-1]+fib[i-2]}

$ print(fib)

[1] 1 1 2 3 5 8 13 21 34 55 89 144

The code is successfully seen running in the following figure.

Graphical user interface, text, application

Description automatically generated

**3.4. With the *mtcars* dataset bundled with R, use *ggplot* to generate a box of miles per gallon (in the variable *mpg*) as a function of the number of gears (in the variable *gear*). Use the fill aesthetic to colour bars by number of gears. (4pt)**

I used the boxplot to create the boxplot using the following x and y variables

x= number of gears (gear)

y=miles per gallon(mpg)

I generated two graphs, the first, although filled didn’t distinguish the mpg covered by cars having different number of gears. The box plot is shown below

$ boxplot(mtcars$mpg ~ mtcars$gear, main = "Box Plot of mpg vs gear", xlab = "gear", ylab = "mpg", col = "lightblue")

Chart, box and whisker chart

Description automatically generated

In the subsequent command, I coloured the miles per gallon covered by different number of gears using the different colours. The box plot is illustrated below.

$ boxplot(mtcars$mpg ~ mtcars$gear, main = "Box Plot of mpg vs gear", xlab = "gear", ylab = "mpg", col=c("lightblue","red","green"))

mtcars$mpg <- as.factor(mtcars$mpg)

**head**(mpg)

ggplot(data=mtcars, aes(x=gear, y=as.factor(mpg), col= c("lightblue","red","green")) + geom\_boxplot()

Chart, box and whisker chart

Description automatically generated

**3.5. Using the *cars* dataset and the function *lm*, fit a linear relationship between *speed* and breaking distance in the variable *distance*. What are the fitted slope and intercept of the line, and their standard errors? What are the units used for the variables in the dataset? (4pt)**

Using the following commands to create a model (mod) we find a linear relationship between speed and breaking distance in the cars dataset.

$ mod = lm(formula = dist ~ speed, data = cars)

$ mod

$ summary(mod)

The intercept of this line (coefficient intercept) is -17.5791, which means that if for instance a car is moving a 0 mph, it would have moved -17.579 feet before coming to a stop.

The fitted slope in our model is 3.9324, which means that for every 1 mph increase in the speed of car, the distance required to stop the car increases by 3.9324 feet.

The standard error for intercept is 6.7584 whilst for the fitted slope is 0.4155.

The distance is measured in feet, whilst the speed is measured in miles per hour (mph).

Please see the below image showing the results of the above code

Graphical user interface, text, application

Description automatically generated

**3.6. Use *ggplot* to plot the data points from Task 6 and the linear fit. (4pt)**

The following code is used to create a ggplot of dist and speed and the linear fit.

This uses the data cars, with speed as the x variable and the braking distance as the y variable.

$ ggplot(cars, aes(x=speed, y=dist)) + geom\_point() + geom\_smooth(method="lm")

Please see the following plot that is generated using the above code.

Chart, scatter chart

Description automatically generated

**3.7. Again using the cars dataset, now use linear regression (*lm*) to estimate the average reaction time for the driver to start breaking (in seconds). To simplify matters you may assume that once breaking commences, breaking distance is proportional to the square of the speed. Explain the steps in your analysis. Do you get reasonable results? Finally, use *ggplot* to plot the data points and the fitted relationship. (9pt)**

Assuming that the breaking distance is proportional to the square of the speed, the above formula will be modified to generate a new model (mod1)

The speed+I(speed^2) equation shows the above generalization.

$ mod1 = lm(formula = dist ~ speed + I(speed^2), data = cars)

$ mod1

The coefficient intercept is calculated as 2.47013.

However, I notice that the average reaction time in the question is in *seconds*, hence we need to convert the speed (mph) into feet/sec, using the conversion that 1mile=5280 feet and 1hour=3600 sec. This will be done using the following code.

#Conversion of mph into feet/sec

$ cars$speed.feetsec <- (cars$speed/3600)\*5280

mod2 <- lm(dist ~ 0 + speed.feetsec + I(speed.feetsec^2), data=cars)

summary(mod2)

**Results:**

The average reaction time is calculated as 0.845 sec.

The mean reaction time to visual stimuli is reported to be around 180-200sec and the auditory stimuli is 140-160msec (1,2). The reaction time does appear longer but there are other confounders that need to be considered like the make of the cars and the effectiveness of the braking system. Depending on these variables, the average reaction time may seem reasonable.

Moreover, the t-value shown is far away from 0.

Assuming a p-value of <0.05 is a good cut-off point, the Pr(>|t|) is shown to be 0.03 and 0.003 and is marked with one and two asterisk respectively which shows that it is significant (i.e.,0.05 and 0.01).

Lastly, the multiple R-squared value is 0.913 which means the 91.3% of variance in the data will be caught by this model.

**ggplot**

The ggplot can be ploted using the following code. This is different to the prior code, as the prior one assumed a relationship using formula using formula 'y ~ x'. However, in this we have assumed that breaking distance is proportional to the square of the speed hence the formula applied will be (formula = "y ~ 0 + x + I(x^2))

$ ggplot(cars, aes(x=speed.feetsec, y=dist)) + geom\_point() + geom\_smooth(method="lm", formula = "y ~ 0 + x + I(x^2)")

This generates the following plot.

Chart, scatter chart

Description automatically generated

**Reference:**

(1) Barrett BT, Cruickshank AG, Flavell JC, et al. Faster visual reaction times in elite athletes are not linked to better gaze stability. *Sci Rep*. 2020;10(1):13216. Published 2020 Aug 6. doi:10.1038/s41598-020-69975-z

(2) Jain A, Bansal R, Kumar A, Singh KD. A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students. *Int J Appl Basic Med Res*. 2015;5(2):124-127. doi:10.4103/2229-516X.157168