############### HOMEWORK 1, ISEN 613, DUE FRI, SEPT 6, 2019, 17:00PM ###############

# YOUR NAME: Joel Dsouza

# RULES:

#

# 1) You can discuss the homework with each other in general terms,

# but you must write your own solutions and not copy from anyone.

#

# 2) Work your way through the R intro and the ISLR Section 2.3.

#

# 3) Edit your answers into this file following each "ANSWER:".

#

# 4) Post your solutions to e-campus

#

# The solutions file must have extension '.txt', NOT '.doc'.

# The filename should have this format: LastName-FirstName-hw01.txt

# An example would be: Eksin-Ceyhun-hw01.txt

#

# 5) Essential criteria for grading, in this order of importance:

# - Generalizability of the solution: Generalizability refers to how

# difficult it would be to modify the code to work for larger or

# slightly different problems.

# - Conciseness: Among two solutions, usually the shorter is preferable.

# PROBLEM 0:

# Set yourself up with R on your computer.

# Follow the instructions on Lecture notes.

# PROBLEM 1: (1pt)

# What happens in this expression?

1:10 - 0:1

# ANSWER:

The output is as below:

**[1] 1 1 3 3 5 5 7 7 9 9**

This means that every odd term of 1:10 is subtracted by 0 and every even term is subtracted by 1.

# PROBLEM 2: (1pt)

# Create a vector consisting of the numbers 1 to N, where 1 appears

# once, 2 appears twice, 3 appears 3 times,...

# Show results for N=5.

# ANSWER:

**n <- as.numeric(readline(prompt="Enter a number: "))** *#run this part first*

**m<-NULL** *#run this part of the code after input*

**for(i in 1:n)**

**{**

**m<-append(m,rep(i,(i:i)))**

**}**

**print(m)**

# PROBLEM 3: (2pts)

# Data generation and matrix indexing

# (a) Generate a vector with 25 elements and each element independently follows a normal

# distribution (with mean =0 and sd=1);

# ANSWER:

**n <- rnorm(25, mean = 0, sd = 1)**

# (b) Reshape this vector into a 5 by 5 matrix in two ways (arranged by row and column);

# ANSWER:

**b <- matrix(n, 5, 5)**

**b <- matrix(n, 5, 5, byrow = T)**

# (c) Similarly, generate another vector with 100 elements and plot its histogram.

# ANSWER:

**n <- rnorm(100, mean = 0, sd = 1)**

**hist(n)**

# (d) Explain the plots in your own words.

# ANSWER:

**The histogram forms a bell curve as per the normal distribution and peaks at 0 since that was the mean set by me.**

# PROBLEM 4: (2pts)

# (a) Upload the Auto data set, which is in the ISLR library. Understand information about this data

# set by one of the ways we introduced in class (like â€œhelp(Auto)â€ and names(Auto))

# ANSWER:

**library(ISLR)**

**help("Auto")**

**names(Auto)**

**head(Auto)**

**tail(Auto)**

# (b) Make a scatterplot between every pair of the following variables (try to plot all scatterplots

# in one figure; hint: use pairs() command): â€œmpgâ€, â€œdisplacementâ€, â€œhorsepowerâ€, â€œweightâ€,

# â€œaccelerationâ€. By observing the plots, do you think the two variables in each scatterplot are

# correlated? If so, how?

# ANSWER:

**pairs(~mpg+displacement+horsepower+weight+acceleration,data=Auto, main="Auto Scatterplot Matrix")**

If two variables are following a trend(upwards or downwards) it means they are somehow correlated. This means that if one variable is increasing, we can infer that the other variable will increase too. For example, we see that if we increase the horsepower of a vehicle, we see greater displacements observed. Similarly, we can also observe negative correlation like in the example of weight and mpg. Lesser the weight, greater is the mileage observed.

# (c) Draw a line on the scatterplot of mpg vs. horsepower to represent relationship between the two

# variables.

# ANSWER:

**plot(mpg, horsepower, main="mpg vs horsepower",xlab="Miles Per Gallon ", ylab="Horsepower ")**

**abline(lm(horsepower ~ mpg, data = Auto), col = "blue")**

# (d) Is there a better way to represent their relationship rather than the linear model you just

# drew? (No need to use a mathematical formula. Just draw something on the figure)

# ANSWER:

**scatter.smooth(x=Auto$horsepower,y=Auto$mpg,main="mpg vs horsepower")**

This graph shows that the data is non linear and would be better fit with a non linear model to understand the data better.

As seen from the figure above, a linear model does not represent the data in a good way. Although it is simple and fast to model the data in a linear fashion, a slightly non linear model would be able to represent the data in a better way.

# PROBLEM 5: (2pts)

# Create a 6x3 matrix named 'dat' with

# row names "Adam", "Anna", "Bill", "Berta", "Chris", "Cindy".

# and column names "Age", "Gender", "Height".

# Fill the columns with realistically looking random numbers

# so someone could actually believe they are real data.

# Use numeric codes 0 and 1 for gender.

# Assume the ages to be uniformly distributed over 18:24.

# Assume men's mean height is 68in and women's 4in less,

# but both heights have the same sdev.

# Finally assume normal distributions for heights,

# except the numbers should be whole inches to look realistic.

# ANSWER:

**boys <- cbind(sample(18:24, 3, replace = TRUE),1,round(rnorm(3,68,0.5),2))**

**girls <- cbind(sample(18:24, 3, replace = TRUE),0,round(rnorm(3,64,0.5),2))**

**colnames(boys) <- c("Age", "Gender", "Height")**

**colnames(girls) <- c("Age", "Gender", "Height")**

**rownames(boys) <- c("Adam","Bill", "Chris")**

**rownames(girls) <- c("Anna", "Berta", "Cindy")**

**dat <- rbind(boys,girls)**

Output:

Age Gender Height

Adam 22 1 67.47

Bill 20 1 67.73

Chris 24 1 68.12

Anna 24 0 64.42

Berta 20 0 63.82

Cindy 20 0 64.41

# BONUS QUESTION: (1 extra pt)

# Create an artificial data matrix with 100000 rows and

# columns "Gender", "Age", "Graduated".

# Assume P(female)=0.57 (code 1) and P(male)=0.43 (code 0).

# Assume Age is uniformly distributed over 18:30 (integers).

# Assume Graduated is a binary variable with codes 0/1 (no/yes).

# Assume further the probabilities of having graduated are

# a function of Age as follows: 0,.01,.1,.2,.7,.95,.99,.99,...

# for ages 18 and up.

# (Hint: To fill 'Graduated', create first a column 'prob' that for

# each individual contains the probability of having graduated

# dependent on 'Age'. This is the tricky part.

# Then set 'Graduated' to runif(100000)<prob.)

# Finally, using the function 'table()', form frequency tables of

# - Graduated,

# - Gender x Age,

# - Gender x Age x Graduated, and

# - Age, but only for the Graduates.

# Check whether the output of the 'table' function is of type 'array'.

# ANSWER:

gender<-rbinom(100000, size = 1, prob = 0.57)

age <- round(runif(100000,min=18,max=30))

x=c(18:30)

y=c(0,0.01,0.1,0.2,0.7,0.95,0.99,0.99,0.99,0.99,0.99,0.99,0.99)

z<-cbind(x,y)

# BONUS QUESTION #2: (1 extra pt)

# Create the following matrix elegantly:

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]

[1,] 0 1 0 1 0 1 0 1 0 1

[2,] 1 0 1 0 1 0 1 0 1 0

[3,] 0 1 0 1 0 1 0 1 0 1

[4,] 1 0 1 0 1 0 1 0 1 0

[5,] 0 1 0 1 0 1 0 1 0 1

[6,] 1 0 1 0 1 0 1 0 1 0

[7,] 0 1 0 1 0 1 0 1 0 1

[8,] 1 0 1 0 1 0 1 0 1 0

[9,] 0 1 0 1 0 1 0 1 0 1

[10,] 1 0 1 0 1 0 1 0 1 0

# ANSWER:

**n=10**

**if(n%%2==0)**

**{**

**m<-matrix(0:1,n+1,n)**

**m<-m[-(n+1),]**

**}**

**if(n%%2==1)**

**{**

**m<-matrix(0:1,n,n+1)**

**m<-m[,-(n+1)]**

**}**

**m**

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]

[1,] 0 1 0 1 0 1 0 1 0 1

[2,] 1 0 1 0 1 0 1 0 1 0

[3,] 0 1 0 1 0 1 0 1 0 1

[4,] 1 0 1 0 1 0 1 0 1 0

[5,] 0 1 0 1 0 1 0 1 0 1

[6,] 1 0 1 0 1 0 1 0 1 0

[7,] 0 1 0 1 0 1 0 1 0 1

[8,] 1 0 1 0 1 0 1 0 1 0

[9,] 0 1 0 1 0 1 0 1 0 1

[10,] 1 0 1 0 1 0 1 0 1 0

## SOME GENERAL COMMENTS AND RECOMMENDATIONS:

##

## - The temptation is often to handcraft a solution if there are

## only few repetitions involved. Try to avoid such

## handcrafting so that your solutions generalize well if the problems are enlarged.

##

## - Try to avoid repeating constants such as '100000'. Think ahead

## for the situations that you have to change the size of the problem.

## Assign constants to variables, and from then on refer to the

## variable, not the constant. This also forces you to think general.

##

## - R has some powerful functions that allow us to do some combinatorial

## problems simply. Examples are:

## \* 'rep()' in its various forms, such as rep(1:3,5) or rep(11:13,1:3).

## \* using vectors for mapping, such as c("red","green","blue")[c(1,3,2,2,1,3)]

## and reverse c(red=1, green=2, blue=3)[c("red","red","blue","green","blue")]