**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

**ITA 0443 - STATISTICS WITH R PROGRAMMING FOR REAL TIME PROBLEM**

**DAY 3 – LAB ASSESSMENT**

**Reg No:192124095**

**Name:K.Abdul Masood**

1. (i) Write a function in R programming to print generate Fibonacci sequence using    
   Recursion in R.

Program:

recur\_fibo <- function(n) {

if(n <= 1) {

return(n)

} else {

return(recur\_fibo(n-1) + recur\_fibo(n-2))

}

}

nterms = as.integer(readline(prompt="How many terms? "))

if(nterms <= 0) {

print("Plese enter a positive integer")

} else {

print("Fibonacci sequence:")

for(i in 0:(nterms-1)) {

print(recur\_fibo(i))

}

}

Output:

How many elements?9

[1] "Fibonacci sequence:"

[1] 0

[1] 1

[1] 1

[1] 2

[1] 3

[1] 5

[1] 8

[1] 13

[1] 21

(ii) Find sum of natural numbers up-to 10, without formula using loopstatement.

Program:

sum = 0

for (i in 1:10) {

sum = sum + i

}

print(sum)

Output:

[1] 55  
  
(iii) create a vector 1:10 and Find a square of each number and store that in a  
separate list.  
Program:

vector <- 1:10

squared\_vector <- vector^2

print(squared\_vector)

Output:

[1] 1 4 9 16 25 36 49 64 81 100  
  
2.    (motor trend car road test) comprises fuelconsumption, performance and  10 aspects of automobile  
design for 32 automobiles. It comes pre-installed  with  package in R.  
 (i)Find the dimension of the dataset

Program:

data(mtcars)

dim(mtcars)

Output:

[1] 32 11  
(ii)Give the statisticalsummary of the features.

Program:

data(mtcars)

summary(mtcars)

Output:

mpg cyl disp hp drat

Min. :10.40 Min. :4.000 Min. : 71.1 Min. : 52.0 Min. :2.760

1st Qu.:15.43 1st Qu.:4.000 1st Qu.:120.8 1st Qu.: 96.5 1st Qu.:3.080

Median :19.20 Median :6.000 Median :196.3 Median :123.0 Median :3.695

Mean :20.09 Mean :6.188 Mean :230.7 Mean :146.7 Mean :3.597

3rd Qu.:22.80 3rd Qu.:8.000 3rd Qu.:326.0 3rd Qu.:180.0 3rd Qu.:3.920

Max. :33.90 Max. :8.000 Max. :472.0 Max. :335.0 Max. :4.930

wt qsec vs am gear

Min. :1.513 Min. :14.50 Min. :0.0000 Min. :0.0000 Min. :3.000

1st Qu.:2.581 1st Qu.:16.89 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:3.000

Median :3.325 Median :17.71 Median :0.0000 Median :0.0000 Median :4.000

Mean :3.217 Mean :17.85 Mean :0.4375 Mean :0.4062 Mean :3.688

3rd Qu.:3.610 3rd Qu.:18.90 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:4.000

Max. :5.424 Max. :22.90 Max. :1.0000 Max. :1.0000 Max. :5.000

carb

Min. :1.000

1st Qu.:2.000

Median :2.000

Mean :2.812

3rd Qu.:4.000

Max. :8.000   
(iii)Print the categorical features in Dataset

Program;

mtcars$am <- factor(mtcars$am, levels = c(0, 1), labels = c("Automatic", "Manual"))

print(mtcars$am)

Output:

[1] Manual Manual Manual Automatic Automatic Automatic Automatic Automatic

[9] Automatic Automatic Automatic Automatic Automatic Automatic Automatic Automatic

[17] Automatic Manual Manual Manual Automatic Automatic Automatic Automatic

[25] Automatic Manual Manual Manual Manual Manual Manual Manual

Levels: Automatic Manual  
(iv)Find the average weight(wt) grouped byEngine shape(vs)

Program:

data(mtcars)

aggregate(wt ~ vs, data = mtcars, mean)

Output:

vs wt

1 0 3.688556

2 1 2.611286  
(v)Find the largest and smallestvalue of the variable weight with respect to Engine shape

Program:

data(mtcars)

aggregate(wt ~ vs, data = mtcars, range)

Output:

vs wt.1 wt.2

1 0 2.140 5.424

2 1 1.513 3.460  
  
3.Use ggplotpackage to plot below EDA questions label the plot accordingly  
(i)Createweight(wt) vs displacement(disp) scatter plot factor by  Engine Shape(vs)

Program:

library(ggplot2)

df <- data.frame(wt = c(3, 4, 5, 6, 7, 8),

disp = c(100, 200, 150, 300, 250, 400),

vs = c("V", "V", "S", "S", "V", "S"))

ggplot(df, aes(x = wt, y = disp, color = "red")) +

geom\_point() +

ggtitle("Weight vs Displacement Scatter Plot Faceted by Engine Shape") +

xlab("Weight (wt)") +

ylab("Displacement (disp)") +

scale\_color\_discrete(name = "Engine Shape (vs)")  
(ii) Create horsepower(hp) vs mileage (mgp) scatter plot factor by  Engine Shape(vs)

Program:

library(ggplot2)

df <- data.frame(hp = c(100, 150, 200, 250, 300, 350),

mgp = c(20, 18, 22, 21, 19, 23),

vs = c("V", "V", "S", "S", "V", "S"))

ggplot(df, aes(x = hp, y = mgp, color = vs)) +

geom\_point() +

ggtitle("Horsepower vs Mileage Scatter Plot Faceted by Engine Shape") +

xlab("Horsepower (hp)") +

ylab("Mileage (mgp)") +

scale\_color\_discrete(name = "Engine Shape (vs)")  
(iv)Inabove plot , Separatecolumns according to cylinders(cyl) size

Program:

library(ggplot2)

df <- data.frame(hp = c(100, 150, 200, 250, 300, 350),

mgp = c(20, 18, 22, 21, 19, 23),

vs = c("V", "V", "S", "S", "V", "S"),

cyl = c(4, 6, 8, 4, 6, 8))

ggplot(df, aes(x = hp, y = mgp, color = vs)) +

geom\_point(aes(shape = as.factor(cyl))) +

ggtitle("Horsepower vs Mileage Scatter Plot Faceted by Engine Shape and Cylinder Size") +

xlab("Horsepower (hp)") +

ylab("Mileage (mgp)") +

scale\_shape\_discrete(name = "Cylinder Size (cyl)")  
(v) Create histogram plot for horsepower (hp)with bin-width size of 5

Program:

library(ggplot2)

df <- data.frame(hp = c(100, 150, 200, 250, 300, 350, 400, 450, 500, 550))

ggplot(df, aes(x = hp)) +

geom\_histogram(binwidth = 5) +

ggtitle("Horsepower Histogram with Bin-Width of 5") +

xlab("Horsepower (hp)") +

ylab("Frequency")  
  
  
  4. PerformingLogistic regression on dataset to predict the cars Engine shape(vs) .  
(i)Do the EDAanalysis and find the features which is impact the Engine shape and use thisfor model.  
(ii) Split the dataset randomly with 80:20 ration to create train and test dataset and create  
logistic model  
(iii)Create theConfusion matrix among prediction and test data.  
  
5. (I) Write R Program to create 15 x15 matrix filled with random numbers between -10 to 10, numbers can repeat. set random seed value to 328.

Program:

set.seed(328)

matrix <- matrix(runif(225, min = -10, max = 10), nrow = 15, ncol = 15)

print(matrix)

Output:

[,1] [,2] [,3] [,4] [,5] [,6]

[1,] 4.650193 8.5982745 -3.4791402 -0.9142146 1.4040338 -9.073830

[2,] 5.601852 6.4087972 3.4444817 9.0329200 3.1132033 -2.539307

[3,] -6.086451 -5.1788499 6.4367553 -8.4596305 -0.7715112 -3.913295

[4,] 6.433422 7.8384104 -8.7492383 1.4152955 -6.0152877 6.185320

[5,] -2.468834 9.0066129 -0.3196906 3.7282971 4.4253976 -1.610087

[6,] 9.778266 1.2960666 -9.1130314 -5.1491268 7.2769148 -9.877884

[7,] -5.717901 -9.0550294 -7.7964704 -6.8597079 -9.5029370 6.250492

[8,] 4.567306 0.6858991 -8.5369708 -8.8627821 -3.4467673 -4.356717

[9,] 8.818495 9.7029622 8.2729891 8.5557580 -3.7369353 -3.566020

[10,] -2.018106 7.3818216 -5.1239983 -3.1757205 -3.7975630 -9.436655

[11,] 5.414889 -7.0444924 7.7643401 7.4922010 6.6073701 -9.558082

[12,] -4.013556 2.6575028 9.9340655 8.3782684 -4.8960865 8.595799

[13,] -9.099600 8.9458338 9.7816288 -7.1914213 5.7198953 4.534948

[14,] 3.426013 5.1860794 2.8406790 -9.3579626 4.2863075 8.670566

[15,] 1.237188 -3.3426812 5.9635083 -3.1029011 -1.2034544 -9.727162

[,7] [,8] [,9] [,10] [,11] [,12]

[1,] 7.622729 6.32264633 9.1713005 1.27882035 -6.8959686 3.6382018

[2,] -5.609092 9.98325313 -2.9076885 6.86641101 -1.5085440 6.3824623

[3,] -6.907849 -1.30280815 9.3670450 6.16947972 9.5761587 -4.9441550

[4,] 1.160646 -9.19050957 -8.1293403 8.36681425 2.9930943 3.6809806

[5,] 6.796707 -8.37721239 -4.9090626 1.50736205 -2.2909891 -1.4709756

[6,] -3.176692 0.74530938 7.5852172 4.00517530 -0.5905598 4.2546398

[7,] 4.651562 1.27931052 0.3400263 -4.36319497 1.5321313 -1.0752759

[8,] 4.653412 4.62269311 -8.3835656 -6.07939875 9.0138266 -4.8739322

[9,] 7.290522 1.82961073 -6.8288456 8.42089997 9.7081804 -6.0002708

[10,] -5.156104 9.45244279 8.0697486 6.54147158 -7.2315279 -7.7270988

[11,] -3.138002 5.84233446 -5.8420461 2.29249234 8.8216364 -6.2607320

[12,] -5.243082 -8.78539019 -9.6810421 2.28374493 0.3615469 -0.5569335

[13,] 7.836100 0.04565765 -3.6154089 5.87754068 -9.9872216 8.9872203

[14,] -6.149255 8.91567017 -3.0722550 0.06503948 5.8774444 6.6048796

[15,] 7.142518 5.18809035 1.1067851 4.05952344 3.9487658 1.1340867

[,13] [,14] [,15]

[1,] -2.037237 -5.3329789 -5.7992909

[2,] 9.599075 9.5730430 7.9282997

[3,] -8.494260 4.5774375 4.1045475

[4,] -3.452614 -4.2174191 -8.5633430

[5,] -7.096982 3.2529362 -0.1800249

[6,] 3.057624 7.5404930 2.4680458

[7,] -9.799904 -6.3983528 -4.7590186

[8,] -6.846358 0.6107489 6.6193893

[9,] -3.922530 3.7071467 -6.2693035

[10,] -1.227993 5.7452154 -8.5353731

[11,] -9.233761 2.8705484 -6.1670496

[12,] -6.110154 1.9214516 -5.7929322

[13,] -0.287593 9.7748768 0.5345717

[14,] 7.154660 0.4255384 5.5250643

[15,] -6.315406 3.8494742 7.6948237

    (ii)Write R Program to display Lower Diagonal and upper Diagonal matrix .

Program:

set.seed(328)

matrix <- matrix(runif(25, min = -10, max = 10), nrow = 5, ncol = 5)

print("Original matrix:")

print(matrix)

print("Lower diagonal:")

print(lower.tri(matrix))

print("Upper diagonal:")

print(upper.tri(matrix))

Output:

"Original matrix:"

[,1] [,2] [,3] [,4] [,5]

[1,] 4.650193 9.778266 5.414889 8.598274 1.2960666

[2,] 5.601852 -5.717901 -4.013556 6.408797 -9.0550294

[3,] -6.086451 4.567306 -9.099600 -5.178850 0.6858991

[4,] 6.433422 8.818495 3.426013 7.838410 9.7029622

[5,] -2.468834 -2.018106 1.237188 9.006613 7.3818216

[1] "Lower diagonal:"

[,1] [,2] [,3] [,4] [,5]

[1,] FALSE FALSE FALSE FALSE FALSE

[2,] TRUE FALSE FALSE FALSE FALSE

[3,] TRUE TRUE FALSE FALSE FALSE

[4,] TRUE TRUE TRUE FALSE FALSE

[5,] TRUE TRUE TRUE TRUE FALSE

[1] "Upper diagonal:"

[,1] [,2] [,3] [,4] [,5]

[1,] FALSE TRUE TRUE TRUE TRUE

[2,] FALSE FALSE TRUE TRUE TRUE

[3,] FALSE FALSE FALSE TRUE TRUE

[4,] FALSE FALSE FALSE FALSE TRUE

[5,] FALSE FALSE FALSE FALSE FALSE

   (iii)Write R Program to count 0's in the matrix and check the matrix is sparse matrix or not.

Program:

set.seed(328)

matrix <- matrix(runif(25, min = 0, max = 1), nrow = 5, ncol = 5)

print("Original matrix:")

print(matrix)

zeros <- sum(matrix == 0)

if(zeros / (nrow(matrix) \* ncol(matrix)) >= 0.5) {

print("The matrix is a sparse matrix")

} else {

print("The matrix is not a sparse matrix")

}

print(paste("Number of zeros in the matrix:", zeros))

Output:

[1] "Original matrix:"

[,1] [,2] [,3] [,4] [,5]

[1,] 0.7325096 0.9889133 0.77074446 0.9299137 0.56480333

[2,] 0.7800926 0.2141050 0.29932221 0.8204399 0.04724853

[3,] 0.1956774 0.7283653 0.04501999 0.2410575 0.53429495

[4,] 0.8216711 0.9409247 0.67130067 0.8919205 0.98514811

[5,] 0.3765583 0.3990947 0.56185941 0.9503306 0.86909108

[1] "The matrix is not a sparse matrix"

[1] "Number of zeros in the matrix: 0"

   (iv) Write R code to remove outliers. Here the outliers are negative numbers. replace the negative values with positive values.

Program:

numbers <- c(-3, 5, 2, -7, 6, -4, 9, -1, 8, -2)

print("Original vector:")

print(numbers)

numbers[numbers < 0] <- abs(numbers[numbers < 0])

print("Updated vector:")

print(numbers)

Output:

[1] "Original vector:"

[1] -3 5 2 -7 6 -4 9 -1 8 -2

[1] "Updated vector:"

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

   (v) Find the mean median and mode of the values corresponding to column

Program:

df <- data.frame(col1 = c(2, 3, 5, 1, 7, 4, 6, 8, 9, 10),

col2 = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))

print("Original data frame:")

print(df)

mean\_col1 <- mean(df$col1)

median\_col1 <- median(df$col1)

mode\_col1 <- names(sort(-table(df$col1)))[1]

print(paste("Mean of column 1:", mean\_col1))

print(paste("Median of column 1:", median\_col1))

print(paste("Mode of column 1:", mode\_col1))

Output:

[1] "Original data frame:"

col1 col2

1 2 1

2 3 2

3 5 3

4 1 4

5 7 5

6 4 6

7 6 7

8 8 8

9 9 9

10 10 10

[1] "Mean of column 1: 5.5"

[1] "Median of column 1: 5.5"

[1] "Mode of column 1: 1"

   (vi)Find the mean median and mode of the values corresponding to row

Program:

df <- data.frame(col1 = c(2, 3, 5, 1, 7, 4, 6, 8, 9, 10),

col2 = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))

print("Original data frame:")

print(df)

mean\_row1 <- mean(df$row1)

median\_row1 <- median(df$row1)

mode\_row1 <- names(sort(-table(df$row1)))[1]

print(paste("Mean of row1:", mean\_row1))

print(paste("Median of row 1:", median\_row1))

print(paste("Mode of row 1:", mode\_row1))

Output:

[1] "Mean of row1: NA"

[1] "Median of row1:NA"

[1] "Mode of row1: NA"