1. Write a R program to

• Create two vectors and perform arithmetic operations.

• Modify a specific element in a vector using subscript operator, logical indexing and function??????????????

v1<-c(11,22,33,44,55,66)

v2<-c(55,44,34,22,21,23)

v1

v2

#arithmetic

v3<-v1+v2

v3

#subscript op

v1[2]<-10

v1

#logical indexing

v1[v1>55]<-0

v1

#c function

v\_1<-v2[c(1,2,3)]

v\_1

...................................................................................................

2. Build a logistic regression model to predict the probability of a customer defaulting based on the average balance carried by the customer using credit dataset.

library(tidyverse)

library(modelr)

library(broom)

#Install ISLR Package

install.packages('ISLR')

#Load ISLR Package

library('ISLR')

(mydata <- as\_tibble(ISLR::Default))

#Checking for NA values

sum(is.na(mydata))

#Creating the Training and Testing data set

sample <- sample(c(TRUE, FALSE), nrow(mydata), replace = T, prob = c(0.6,0.4))

train <- mydata[sample, ]

test <- mydata[!sample, ]

logmodel <- glm(default ~ balance, family = "binomial", data = train)

mydata %>%

mutate(prob = ifelse(default == "Yes", 1, 0)) %>%

ggplot(aes(balance, prob)) +

geom\_point(alpha = .15) +

geom\_smooth(method = "glm", method.args = list(family = "binomial")) +

ggtitle("Logistic regression model fit") +

xlab("Balance") +

ylab("Probability of Default")

#Summary of the Logistic Regression Model

summary(logmodel)

..............................................................................................................

3. Write a R program to :

• Find the minimum and maximum value in a Vector.

• Concatenate two matrices of same column but different rows.

• Add 10 to each element of the first vector in a given list

• Draw a boxplot with dots for all elements of a given list.

#min and max of vector

v<-c(122,543,221,33,342)

max(v)

min(v)

#Concatenate two matrices of same column but different rows.

matrix1 <- matrix(1:6, nrow = 2, ncol = 3)

matrix2 <- matrix(7:12, nrow = 2, ncol = 3)

print(matrix1)

print(matrix2)

concatenated\_matrix <- rbind(matrix1, matrix2)

print(concatenated\_matrix)

# Add 10 to each element of the first vector in a given list

my\_list <- list(vec1 = c(1, 2, 3, 4, 5), vec2 = c(6, 7, 8, 9, 10))

my\_list$vec1 <- my\_list$vec1 + 10

print(my\_list)

un\_list<-unlist(my\_list)

un\_list

boxplot(un\_list)

points(un\_list, pch = 16, col = "red")

............................................................................................

4. Construct a R program to identify frequent pattern using the association rule for the Groceries dataset.

## Example: Identifying Frequently-Purchased Groceries ----

# load the grocery data into a sparse matrix

library(arules)

library(Matrix)

groceries<- read.transactions("D://DRIVERS/Dataset/groceries.csv", sep = ",")

summary(groceries)

# look at the first five transactions

inspect(groceries[1:5])

# examine the frequency of items

itemFrequency(groceries[, 1:3])

# plot the frequency of items

itemFrequencyPlot(groceries, support = 0.1)

itemFrequencyPlot(groceries, topN = 20)

# a visualization of the sparse matrix for the first five transactions

image(groceries[1:5])

# visualization of a random sample of 100 transactions

image(sample(groceries, 100))

## Step 3: Training a model on the data ----

library(arules)

# default settings result in zero rules learned

apriori(groceries)

# set better support and confidence levels to learn more rules

groceryrules<- apriori(groceries, parameter = list(support = 0.006, confidence = 0.25, minlen = 2))

groceryrules

## Step 4: Evaluating model performance ----

# summary of grocery association rules

summary(groceryrules)

# look at the first three rules

inspect(groceryrules[1:3])

## Step 5: Improving model performance ----

# sorting grocery rules by lift

inspect(sort(groceryrules, by = "lift")[1:5])

# finding subsets of rules containing any berry items

berryrules<- subset(groceryrules, items %in% "berries")

inspect(berryrules)

# writing the rules to a CSV file

write(groceryrules, file = "groceryrules.csv",

sep = ",", quote = TRUE, row.names = FALSE)

# converting the rule set to a data frame

groceryrules\_df<- as(groceryrules, "data.frame")

str(groceryrules\_df)

...............................................................................................................................

5. Write a R program to:

• Create two matrices and define the column and row names.

• Access the element at 3rd column and 1st row and arrange them sequentially by column

• Add and Subtract two matrices

m1<- matrix(1:6, nrow = 2, ncol = 3)

colnames(m1) <- c("Column1", "Column2", "Column3")

rownames(m1) <- c("Row1", "Row2")

m1

m2<-matrix(7:12, nrow = 2, ncol = 3)

colnames(m2) <- c("ColumnA", "ColumnB", "ColumnC")

rownames(m2) <- c("RowA", "RowB")

m2

element <- m1[1, 3]

arranged\_element <- c(element)

print(arranged\_element)

m1<- matrix(1:6, nrow = 2, ncol = 3)

m2<- matrix(7:12, nrow = 2, ncol = 3)

add\_result <- m1 + m2

add\_result

# Subtract the matrices

subtract\_result <- m1 - m2

subtract\_result

................................................

6. Write a R program to calculate the following for the red wine dataset:

• Find min, max and range from dataset

• Find variance, quartile and standard deviation

• Find IQR and summarize the dataset.

red<-read.csv("D://DRIVERS/Dataset/redwine.csv")

red

min\_red<-min(red$sulphates)

min\_red

max\_red<-max(red$sulphates)

max\_red

r1<-max\_red-min\_red

r1

ran<-range(red$sulphates)

ran

v1<-var(red$sulphates)

v1

q1<-quantile(red$sulphates)

q1

s1<-sd(red$sulphates)

s1

iqr1<-IQR(red$sulphates)

iqr1

su1<-summary(red$sulphates)

su1

summary(red)

...............................................................

7. Write a R program to perform the following using used car dataset:

• Find min, max and range from dataset

• Find variance, quartile, and standard deviation

• Find IQR and summarize the dataset

used<-read.csv("D://DRIVERS/Dataset/usedcars.csv")

used

min\_car<-min(used$price)

min\_car

max\_car<-max(used$price)

max\_car

r<-max\_car-min\_car

r

ra<-range(used$price)

ra

v<-var(used$price)

v

q<-quantile(used$price)

q

s<-sd(used$price)

s

iqr<-IQR(used$price)

iqr

su<-summary(used$price)

su

summary(used)

..............................................................................................................................................

9. Create a Data frame for Students mark list with 20 observations for 5 numerical variables and perform the following:

• Find min, max and range from dataset

• Find variance, quartile and standard deviation

• Find IQR and summarize the dataset

• Visualize using histogram and 3D plot

name<-sample(c("clement","raj","shasha","samyu"),20,replace=TRUE)

id<-1:20

m1<-sample(20:100,20,replace=TRUE)

m2<-sample(20:100,20,replace=TRUE)

m3<-sample(20:100,20,replace=TRUE)

m4<-sample(20:100,20,replace=TRUE)

m5<-sample(20:100,20,replace=TRUE)

stud<-data.frame(name,id,m1,m2,m3,m4,m5)

stud

mean1<-mean(stud$m1)

mean1

mean1<-mean(stud$m1)

med<-median(stud$m1)

mode1<-mode(stud$m1)

ran<-range(stud$m1)

mini<-min(stud$m1)

maxi<-max(stud$m1)

med

mode1

ran

mini

maxi

summary(stud)

std<-sd(stud$m3)

variance<-var(stud$m3)

quartile<-quantile(stud$m2)

std

variance

quartile

hist(stud$m1, main ="Used Cars Mileage",

xlab ="m1",

col ="yellow",border="red")

x<-stud$m1

y<-stud$m2

z<-stud$m3

scatterplot3d(y,x,z,

xlab = "m1",

ylab = "m2",

zlab = "m3",

main = "student marks",

color="violet")

..................................................................................

10. Build a Machine Learning model that predicts whether an applicant’s loan can be approved or not based on KNN algorithm for the given credit dataset.

loan<-read.csv(file="D://DRIVERS/Dataset/credit.csv")

loan.subset<-loan[c('Creditability','Account.Balance','Purpose','Instalment.per.cent','Length.of.current.employment','Occupation')]

head(loan.subset)

nor<-function(x)

{

return((x-min(x))/(max(x)-min(x)))

}

loan.subset.n<-as.data.frame(lapply(loan.subset[,-1],nor))

head(loan.subset.n)

library(class)

library(caret)

library(caTools)

set.seed(123)

sample<-sample(1:nrow(loan.subset.n),size = nrow(loan.subset.n)\*0.7,replace = FALSE)

train.loan<-loan.subset[sample,]

test.loan<-loan.subset[-sample,]

train.loan.la<-loan.subset[sample,1]

test.loan.la<-loan.subset[-sample,1]

NROW(train.loan.la)

knn.26<-knn(train = train.loan,test = test.loan,cl=train.loan.la,k=26)

knn.26

acc.26<-100\*sum(test.loan.la==knn.26)/NROW(test.loan.la)

acc.26

table(knn.26,test.loan.la)

confusionMatrix(table(knn.26,test.loan.la))

i=1

k.opt=1

for(i in 1:28)

{

knn.m<-knn(train = train.loan,test = test.loan,cl=train.loan.la,k=i)

k.opt[i]<-100\*sum(test.loan.la==knn.m)/NROW(test.loan.la)

k=i

cat(k,'=',k.opt[i],'')

}

plot(k.opt,type = "b",xlab = "k-value",ylab = "acuuracy")

............................................................................................

11. Create a data frame and Pre-Process the same.

mydata<-read.csv("D://DRIVERS/Dataset/mysampledata.csv")

mydata

#dealing with missing values

mydata$age<-ifelse(is.na(mydata$age),ave(mydata$age,FUN = function(x)

mean(x,na.rm=TRUE)),mydata$age)

mydata

mydata$salary<-ifelse(is.na(mydata$salary),ave(mydata$salary, FUN = function(x)

mean(x,na.rm=TRUE)),mydata$salary)

mydata

mydata$age<-as.numeric(format(round(mydata$age,0)))

#Dealing with categorical Data

mydata$nation<-factor(mydata$nation,levels=c('India','Russia','Germany'),labels=c(1,2,3))

mydata$purchased\_item<-factor(mydata$purchased\_item,levels = c('No','Yes'),labels=c(0,1))

mydata

install.Packages('caTools')

library(caTools)

set.seed(123)

Split<-sample.split(mydata$purchased\_item,SplitRatio=0.8)

training\_set<-subset(mydata,Split==TRUE)

test\_set<-subset(mydata,Split==FALSE)

training\_set[,3:4]<-scale(training\_set[,3:4])

test\_set[,3:4]<-scale(test\_set[,3:4])

...................................................................................

12. Write a R program to:

• Create and sort a Vector in ascending and descending order.

• Create a list of data frames and access each of those data frames from the list.

• Draw a 3D plot for the white wine dataset.

v1<-c(1,33,44,2,4)

v2<-sort(v1)

v2

decs<-sort(v1,decreasing = TRUE)

decs

stu<-data.frame(stud=c("anu","mahisha","aish"),id=c(1,7,4))

sub<-data.frame(sub=c("cse","msc","it"))

mark<-data.frame(mrk=c(450,465,345))

#accessing values in list

stds<-list(stu=stu,sub=sub,mark=mark)

stds$stu

stds$stu$id

stds$stu[2,2]

stds[2]

stds[3]

stds$sub

stds$mark

stds

# Load the white wine dataset (assuming it's already installed)

mywine<-read.csv("D://DRIVERS/Dataset/redwine.csv")

str(mywine)

# Extract the variables from the dataset

x <- sort(mywine$fixed.acidity)

y <- sort(mywine$volatile.acidity)

z <- mywine$alcohol

library(scatterplot3d)

scatterplot3d(y,x,z,

xlab = "Fixed Acidity",

ylab = "Volatile Acidity",

zlab = "Alcohol Content",

main = "Redwine - 3D Plot",

color="violet")

...................................................................................

13. Using Decision tree algorithm, classify the Mushroom data set in order to predict whether a given mushroom is edible or poisonous to human beings.

library(caTools)

library(caret)

library(rpart)

library(rpart.plot)

library(rattle)

mush<-read.csv("D://DRIVERS/Dataset/mushrooms.csv")

str(mush)

summary(mush)

nrow(mush)-sum(complete.cases(mush))

mush$veil\_type<-NULL

mush$veil.type<NULL

mush$veil.class<NULL

table(mush$class,mush$odor)

set.seed(12345)

sample<-sample(1:nrow(mush),size = ceiling(0.80\*nrow(mush)),replace = FALSE)

train<-mush[sample,]

test<-mush[-sample,]

m<-matrix(c(0,1,10,0),byrow = TRUE,nrow = 2)

m

tree<-rpart(class~.,data=train,method = "class",parms=list(loss=m))

tree

rpart.plot(tree,nn=TRUE)

pred<-predict(object = tree,test[-1],type="class")

pred

t<-table(test$class,pred)

t

confusionMatrix(t)

....................................................................................................

14. Create an employee dataset with the following fields (Employee id, name, department, salary, mobile number) and pre-process the same.

# Create sample data

id <- 1:20

dept<-sample(c("it","management","hr","sales"),20,replace=TRUE)

sal = c(30000, 40000, NA, 50000, NA, 60000, 70000, 80000, NA, 90000, 100000,

110000, NA, 120000, 130000, 140000, NA, 150000, 160000, 170000)

mobilenum<-sample(1234678953:98765432163,20,replace=TRUE)

name <- c("Anu", "James", "John", "Emily", "Michael",

"Emma", "Daniel", "Sophia", "David", "Olivia",

"Matthew", "Charlotte", "Andrew", "Ava", "William",

"Amelia", "Ryan", "Mia", "Ethan", "Isabella")

# Create dataframe

mydata <- data.frame(id = id, name = name,sal=sal,dept=dept,mobilenumber=mobilenum)

mydata

# Display the datafram

mydata$sal=ifelse(is.na(mydata$sal),mean(mydata$sal,na.rm=TRUE),mydata$sal)

mydata$sal

mydata

#Dealing With Categorical Data

mydata$dept<-factor(mydata$dept,levels=c('hr','sales','it','management'),labels=c(1,2,3,4))

mydata

install.packages('caTools') #install once

library(caTools) # importing caTools library

set.seed(123)

split <-sample.split(mydata$sal, SplitRatio = 0.8)

training\_set <- subset(mydata, split == TRUE)

test\_set <- subset(mydata, split == FALSE)

training\_set

test\_set

#Scaling the Features

training\_set[3] = scale(training\_set[3])

test\_set[3] = scale(test\_set[3])

training\_set[3]

test\_set[3]

.................................................................................................................

5......Central tendency

usedcars <- read.csv("D://DRIVERS/Dataset/usedcars.csv", stringsAsFactors = FALSE)

str(usedcars)

print(head(usedcars))

max1=max(usedcars$price)

print(max1)

min1=min(usedcars$price)

print(min1)

range=max1-min1

print(range)

r=range(usedcars$price)

print(r)

variance=var(usedcars$price)

print(variance)

std=sd(usedcars$price)

quartiles=quantile(usedcars$price)

print(quartiles)

IQR=IQR(usedcars$price)

print(IQR)

summary=summary(usedcars$price)

print(summary)

summary(usedcars)

...............................................................................................................

4. Visualizing numeric data – scatterplot, boxplot, piechart, histograms

#Boxplot

us\_cars<-read.csv("D://DRIVERS/Dataset/usedcars.csv",stringsAsFactors = FALSE)

str(us\_cars)

boxplot(us\_cars$price,main="Used cars price",ylab="price($)",col="blue",border="red")

boxplot(us\_cars$mileage,main="Used cars mileage",ylab="Odometer(mi..)",col="red",border="brown")

boxplot(us\_cars[,0:6], main ='Used Cars ')

#Bar Plot

barplot(us\_cars$mileage,

main = 'Used Cars',

xlab = 'Mileage', horiz = FALSE,col='blue')

#Histogram

hist(us\_cars$mileage, main ="Used Cars Mileage",

xlab ="Mileage",

col ="yellow",border="red")

hist(us\_cars$price, main ="Used Cars Price",

xlab ="Price",

col ="Green",border="red")

#Scatter Plot

plot(us\_cars$price,us\_cars$mileage,

main ="Scatterplot ",

xlab ="Price",

ylab =" Mielage", pch = 12,col='green')

#Pie

n=12

pie(rep(1,n), col=rainbow(n))

pie(rep(1,n), col=heat.colors(n))

pie(rep(1,n), col=terrain.colors(n))

#3D Graphs

cone <- function(x, y){

sqrt(x ^ 2 + y ^ 2)

}

# prepare variables.

x <- y <- seq(-1, 1, length = 30)

z <- outer(x, y, cone)

# plot the 3D surface

# Adding Titles and Labeling Axes to Plot

persp(x, y, z,

main="Perspective Plot of a Cone",

zlab = "Height",

theta = 30, phi = 15,

col = "green", shade = 0.4)

................................................................................

3.saving ,loading,removing...........

obj1<-c(1:15)

obj2<-FALSE

Obj3<-"Welcome to R Environment"

save.image("ex1.Rdata")

obj1<-c(1:15)

obj2<-FALSE

Obj3<-"Welcome to R Environment"

save(obj1, obj2, file ="tempworkspaceobj.RData")

load("tempworkspaceobj.RData")

.....................................................................................

2.creation of arrays

#Creation of an Array

vector1 <- c(1, 2, 3)

vector2 <- c(10, 15, 3, 11, 16, 12)

result <- array(c(vector1, vector2), dim = c(3, 3, 2))

print(result)

#Naming columns and rows

vector1 <- c(1, 2, 3)

vector2 <- c(10, 15, 3, 11, 16, 12)

column.names <- c("COL1", "COL2", "COL3")

row.names <- c("R1", "R2", "R3")

matrix.names <- c("Matrix.NO1", "Matrix.NO2")

result <- array(c(vector1, vector2), dim = c(3, 3, 2),

dimnames = list(row.names, column.names, matrix.names))

print(result)

#Manipulating array elements

vector1 <- c(1, 2, 3)

vector2 <- c(4, 6, 8, 0, 2, 4)

array1 <- array(c(vector1, vector2), dim = c(3, 3, 2))

vector3 <- c(3, 2, 1)

vector4 <- c(2, 4, 6, 8, 3, 5)

array2 <- array(c(vector3, vector4), dim = c(3, 3, 2))

matrix1 <- array1[,,2]

matrix2 <- array2[,,2]

result <- matrix1 + matrix2

print(result)

result<-matrix1-matrix2

print(result)

result<-matrix1\*matrix2

result<-matrix1/matrix2

#Accessing Array elements

vector1 <- c(1, 2, 3)

vector2 <- c(10, 15, 3, 11, 16, 12)

column.names <- c("COLUMN1", "COLUMN2", "COLUMN3")

row.names <- c("ROW1", "ROW2", "ROW3")

matrix.names <- c("Matrix.NO1", "Matrix.NO2")

# taking vector as input

result <- array(c(vector1, vector2), dim = c(3, 3, 2),

dimnames = list(row.names, column.names, matrix.names))

print(result)

# print third row of second matrix

print(result[3,,2])

#Calculation across array element

vector1 <- c(3, 2, 1)

vector2 <- c(2, 4, 6, 8, 0, 1)

new.array <- array(c(vector1, vector2), dim = c(3, 3, 2))

print(new.array)

# using apply and calculate the sum of rows in matrices

result <- apply(new.array, c(1), sum)

print(result)

#Creation of Matrices:

# Elements are arranged sequentially by row.

M <- matrix(c(3:14), nrow = 4, byrow = TRUE)

print(M)

# Elements are arranged sequentially by column.

N <- matrix(c(3:14), nrow = 4, byrow = FALSE)

print(N)

# Define the column and row names.

rownames = c("row1", "row2", "row3", "row4")

colnames = c("col1", "col2", "col3")

P <- matrix(c(3:14), nrow = 4, byrow = TRUE, dimnames = list(rownames, colnames))

print(P)

#Operations on Matrices

#Matrices Addition

# Creating 1st Matrix

B = matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3)

# Creating 2nd Matrix

C = matrix(c(7, 8, 9, 10, 11, 12), nrow = 2, ncol = 3)

# Getting number of rows and columns

num\_of\_rows = nrow(B)

num\_of\_cols = ncol(B)

# Creating matrix to store results

sum = matrix(, nrow = num\_of\_rows, ncol = num\_of\_cols)

# Printing Original matrices

print(B)

print(C)

# Creating 1st Matrix

B <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3)

# Creating 2nd Matrix

C <- matrix(c(7, 8, 9, 10, 11, 12), nrow = 2, ncol = 3)

# Getting number of rows and columns

num\_of\_rows <- nrow(B)

num\_of\_cols <- ncol(B)

# Creating matrix to store results

diff <- matrix(, nrow = num\_of\_rows, ncol = num\_of\_cols)

# Printing Original matrices

print(B)

print(C)

# Calculating diff of matrices

for(row in 1:num\_of\_rows) {

for(col in 1:num\_of\_cols) {

diff[row, col] <- B[row, col] - C[row, col]

}

}

# Print the difference matrix

print(diff)

# Printing resultant matrix

print(diff)

# Calculating product of matrices

for(row in 1:num\_of\_rows) {

for(col in 1:num\_of\_cols) {

prod[row, col] <- B[row, col] \* C[row, col]

}

}

# Print the product matrix

print(prod)

# Printing resultant matrix

print(prod)

# R program for matrix multiplication

# using '\*' operator

# Creating 1st Matrix

B = matrix(c(1, 2 + 3i, 5.4), nrow = 1, ncol = 3)

# Creating 2nd Matrix

C = matrix(c(2, 1i, 0.1), nrow = 1, ncol = 3)

# Printing the resultant matrix

print (B \* C)

#Matrices Division

# Creating 1st Matrix

B = matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3)

# Creating 2nd Matrix

C = matrix(c(7, 8, 9, 10, 11, 12), nrow = 2, ncol = 3)

# Getting number of rows and columns

num\_of\_rows = nrow(B)

num\_of\_cols = ncol(B)

# Creating matrix to store results

div = matrix(, nrow = num\_of\_rows, ncol = num\_of\_cols)

# Printing Original matrices

print(B)

print(C)

# Calculating product of matrices

for(row in 1:num\_of\_rows)

{

for(col in 1:num\_of\_cols)

{

div[row, col] <- B[row, col] / C[row, col]

}

}

# Printing resultant matrix

print(div)

# Creating 1st Matrix

B = matrix(c(4, 6i, -1), nrow = 1, ncol = 3)

# Creating 2nd Matrix

C = matrix(c(2, 2i, 0), nrow = 1, ncol = 3)

# Printing the resultant matrix

print (B / C)

...........................................................................

1.vectors

#Creating a vector

X <- c(1, 4, 5, 2, 6, 7)

print('using c function')

print(X)

Y <- seq(1, 10, length.out = 5)

print('using seq() function')

print(Y)

Z <- 5:10

print('using colon')

print(Y)

#Accessing vector elements

X <- c(2, 5, 8, 1, 2)

print('using Subscript operator')

print(X[2])

Y <- c(4, 5, 2, 1, 7)

print('using c function')

print(Y[c(4, 1)])

Z <- c(5, 2, 1, 4, 4, 3)

print('Logical indexing')

print(Z[Z>3])

#Modifying a vector

X <- c(2, 5, 1, 7, 8, 2)

# modify a specific element

X[3] <- 11

print('Using subscript operator')

print(X)

# Modify using different logics.

X[X>9] <- 0

print('Logical indexing')

print(X)

# Modify by specifying the position or elements.

X <- X[c(5, 2, 1)]

print('using c function')

print(X)

#Deleting a vector

#Creating a vector

X <- c(5, 2, 1, 6)

# Deleting a vector

X <- NULL

print('Deleted vector')

print(X)

#Arithmetic operations

#Creating Vectors

X <- c(5, 2, 5, 1, 51, 2)

Y <- c(7, 9, 1, 5, 2, 1)

# Addition

Z <- X + Y

print('Addition')

print(Z)

# Subtraction

S <- X - Y

print('Subtraction')

print(S)

# Multiplication

M <- X \* Y

print('Multiplication')

print(M)

# Division

D <- X / Y

print('Division')

print(D)

#Sorting of Vectors

#Creating a Vector

X <- c(5, 2, 5, 1, 51, 2)

# Sort in ascending order

A <- sort(X)

print('sorting done in ascending order')

print(A)

# sort in descending order.

B <- sort(X, decreasing = TRUE)

print('sorting done in descending order')

print(B)

# Creating List

#Creating Vectors

vec1 <- c(1, 2, 3)

vec2 <- c(TRUE, FALSE)

# Creating a list of Vectors

listt = list(vec1, vec2)

# Printing List

print (listt)

#Adding elements to a list

# Creating Vectors

vec1 <- c(1, 2, 3)

vec2 <- c(TRUE, FALSE)

# Creating list of Vectors

lst = list(vec1, vec2)

# Creating a new Vector

vec3 <- c(1 + 3i)

# Adding Vector to list

lst[[3]]<- vec3

# Printing List

print (lst)

# determine the length of list

len <- length(lst)

# Creating new Vector

vec3 <- c(0.5, 2 + 2i)

# Using for loop to add elements

for( i in 1:2)

{

# Adding vec to list

lst[[len + i]]<- vec3

}

print (lst)

#Removing elements from a list

lst[[2]]<-NULL

print ("Modified List")

print (lst)

#Modifying elements in a list

# Creating Vectors

vec1 <- c(1, 2, 3)

vec2 <- c(TRUE, FALSE)

# Creating list of Vectors

lst = list(vec1, vec2)

print ("original list")

print (lst)

# Modifying List element

lst[[2]]<-c("TEACH", "CODING")

print ("Modified List")

print (lst)

#Merging two lists

# R program to merge two lists of Vectors

# Creating 1st list

list\_data1 <- list(c(1:3), c(TRUE, FALSE))

# Creating 2nd list

list\_data2 <- list(c(0.1, 3.4))

print("First List")

print (list\_data1)

print ("Second List")

print (list\_data2)

print("Merged List")

# Merging Lists

merged\_list <- c(list\_data1, list\_data2)

print (merged\_list)

.................................................................................................................................................................................................