**Experiment No:** 1

**Experiment Name:** To perform the basic mathematical operations in R programming.

**Name:**

**Roll No:-**

**R Script:-**

#Assignment

s <- 10

print(s)

#case sensitive

a <- 5

b <- 2

c <- 1

print(a+B+c)

#Arithmetic Operations

#creation of vector

a <- c(10, 20, 30, 40, 50)

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

print(a)

print(b)

#Addition of two vector

Result <- a+b

print(Result)

#Subtraction of two vector

Result <- a-b

print(Result)

#Multiplication of two vector

Result <- a\*b

print(Result)

#Division of two vector

Result <- a/b

print(Result)

#options()

1/7#by default it will show 7 digits output.

options(digits = 3)#by using this it will show only 3 digits after decimal point

1/7

#Miscellaneous Mathematical functions

x<-20

abs(x) #Absolute Value

sqrt(x) #square root

exp(x) #exponential transformation

log(x) #logarithmic transformation

cos(x) #cosine and other trigonometric transformation

#infinite and Nan Number

y<-5

z<-6

ls() #List all object

exists("y") #identify R object with 'y' name

rm(y) #remove object.

rm(y,z) #remove multiple object.

rm(list=ls()) #remove everything on working environment.

**OUTPUT-**

|  |
| --- |
| > #Assignment  > s <- 10  > print(s)  [1] 10  >  > #case sensitive  > a <- 5  > b <- 2  > c <- 1  > print(a+B+c)  Error in print(a + B + c) : object 'B' not found  >  > #Arithmetic Operations  > #creation of vector  >  > a <- c(10, 20, 30, 40, 50)  > b <- c(1, 2, 3, 4, 5)  > print(a)  [1] 10 20 30 40 50  > print(b)  [1] 1 2 3 4 5  >  > #Addition of two vector  > Result <- a+b  > print(Result)  [1] 11 22 33 44 55  >  > #Subtraction of two vector  > Result <- a-b  > print(Result)  [1] 9 18 27 36 45  >  > #Multiplication of two vector  > Result <- a\*b  > print(Result)  [1] 10 40 90 160 250  >  > #Division of two vector  > Result <- a/b  > print(Result)  [1] 10 10 10 10 10  >  > #options()  > 1/7#by default it will show 7 digits output.  [1] 0.1428571  >  > options(digits = 3)#by using this it will show only 3 digits after decimal point  > 1/7  [1] 0.143  >  > #Miscellaneous Mathematical functions  >  > x<-20  > abs(x) #Absolute Value  [1] 20  > sqrt(x) #square root  [1] 4.47  > exp(x) #exponential transformation  [1] 4.85e+08  > log(x) #logarithmic transformation  [1] 3  > cos(x) #cosine and other trigonometric transformation  [1] 0.408  >  > #infinite and Nan Number  > y<-5  > z<-6  >  > ls() #List all object  [1] "a" "b" "c" "Result" "s" "x" "y" "z"  > exists("y") #identify R object with 'y' name  [1] TRUE  > rm(y) #remove object.  > rm(y,z) #remove multiple object.  Warning message:  In rm(y, z) : object 'y' not found  > rm(list=ls()) #remove everything on working environment. |
|  |

**Experiment No:** 2

**Experiment Name:** Write program for creating and manipulating R objects in R-Vectors, Matrices, Array, Dataframes, List.

**Name:**

**Roll No:-**

**Atomic Vector:-**

**Numeric vector: -**

num\_vec<-c(10.1, 10.2, 33.2)

num\_vec

**Integer vector: -**

num<-c(2L,6L,4L,9L)

num

**Character vector: -**

fruits<-c("Mango","apple","papaya")

print(fruits)

**Logical vector: -**

a<-as.integer(20)

b<-as.integer(10)

log\_vec<-c(a<b,b<a,a>b,b>a)

log\_vec

**Operations on Vector: -**

1)combining vectors:

data\_vec<-c(names,num)

data\_vec

**2)Arithmetic operations:**

a<-c(1,3,5,7)

b<-c(2,4,6,8)

a+b

a-b

a\*b

a/b

**3)Logical Index vector:**

z<-c(1,2,3,4,5,6)

z[c(TRUE,FALSE,TRUE,TRUE,FALSE,TRUE)]

**4)Numeric Index: -**

q<-c("shubham","arpita","nishka","gunjan","vaishali","sumit")

q[2]

q[-4]

q[15]

**5)Duplicate Index: -**

q<-c("shubham","arpita","nishka","gunjan","vaishali","sumit")

q[c(2,4,4,3)]

**6)Range Indexes: -**

q<-c("shubham","arpita","nishka","gunjan","vaishali","sumit")

b<-q[2:5]

b

**7)out-of-order Indexes: -**

q<-c("shubham","arpita","nishka","gunjan","vaishali","sumit")

q[c(2,1,3,4,5,6)]

**8)Named vectors members: -**

z=c("Roshani","Kawale")

z

names(z)=c("FirstName","LastName")

z

z["FirstName"]

**OUTPUT: -**

> #Numeric vector: -

> num\_vec<-c(10.1, 10.2, 33.2)

> num\_vec

[1] 10.1 10.2 33.2

>

> #Integer vector: -

> num<-c(2L,6L,4L,9L)

> num

[1] 2 6 4 9

>

> #Character vector: -

> fruits<-c("Mango","apple","papaya")

> print(fruits)

[1] "Mango" "apple" "papaya"

>

> #Logical vector: -

> a<-as.integer(20)

> b<-as.integer(10)

> log\_vec<-c(a<b,b<a,a>b,b>a)

> log\_vec

[1] FALSE TRUE TRUE FALSE

>

> #Operations on Vector: -

> #1)combining vectors:

> data\_vec<-c(names,num)

> data\_vec

[[1]]

function (x) .Primitive("names")

[[2]]

[1] 2

[[3]]

[1] 6

[[4]]

[1] 4

[[5]]

[1] 9

>

> #2)Arithmetic operations:

> a<-c(1,3,5,7)

> b<-c(2,4,6,8)

> a+b

[1] 3 7 11 15

> a-b

[1] -1 -1 -1 -1

> a\*b

[1] 2 12 30 56

> a/b

[1] 0.5000000 0.7500000 0.8333333 0.8750000

>

> #3)Logical Index vector:

> z<-c(1,2,3,4,5,6)

> z[c(TRUE,FALSE,TRUE,TRUE,FALSE,TRUE)]

[1] 1 3 4 6

>

>

> #4)Numeric Index: -

> q<-c("shubham","arpita","nishka","gunjan","vaishali","sumit")

> q[2]

[1] "arpita"

> q[-4]

[1] "shubham" "arpita" "nishka" "vaishali" "sumit"

> q[15]

[1] NA

>

> #5)Duplicate Index: -

> q<-c("shubham","arpita","nishka","gunjan","vaishali","sumit")

> q[c(2,4,4,3)]

[1] "arpita" "gunjan" "gunjan" "nishka"

>

> #6)Range Indexes: -

> q<-c("shubham","arpita","nishka","gunjan","vaishali","sumit")

> b<-q[2:5]

> b

[1] "arpita" "nishka" "gunjan" "vaishali"

>

> #7)out-of-order Indexes: -

> q<-c("shubham","arpita","nishka","gunjan","vaishali","sumit")

> q[c(2,1,3,4,5,6)]

[1] "arpita" "shubham" "nishka" "gunjan" "vaishali" "sumit"

>

> #8)Named vectors members: -

> z=c("Roshani","Kawale")

> z

[1] "Roshani" "Kawale"

> names(z)=c("FirstName","LastName")

> z

FirstName LastName

"Roshani" "Kawale"

> z["FirstName"]

FirstName

"Roshani"

**#creation of matrix:-**

P <- matrix(c(5:16), nrow = 4, byrow = TRUE)

print(P)

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

print(Q)

**#operations on Matrix:-**

**#1)Addition:-**

sum<-P+Q

print(sum)

**#2)Subtraction:-**

sub<-P-Q

print(sub)

**#3)Multiplication(\*):-**

mult<-P\*Q

print(mult)

**#4)Multiplication(by constant):-**

mult<-P\*5

print(mult)

**#5)Division:-**

div<-P/Q

div

**OUTPUT:-**

**#**creation of matrix:-

> P <- matrix(c(5:16), nrow = 4, byrow = TRUE)

> print(P)

[,1] [,2] [,3]

[1,] 5 6 7

[2,] 8 9 10

[3,] 11 12 13

[4,] 14 15 16

>

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

> print(Q)

[,1] [,2] [,3]

[1,] 3 7 11

[2,] 4 8 12

[3,] 5 9 13

[4,] 6 10 14

>

**>** #operations on Matrix:-

> #1)Addition:-

> sum<-P+Q

> print(sum)

[,1] [,2] [,3]

[1,] 8 13 18

[2,] 12 17 22

[3,] 16 21 26

[4,] 20 25 30

> #2)Subtraction:-

> sub<-P-Q

> print(sub)

[,1] [,2] [,3]

[1,] 2 -1 -4

[2,] 4 1 -2

[3,] 6 3 0

[4,] 8 5 2

> #3)Multiplication(\*):-

> mult<-P\*Q

> print(mult)

[,1] [,2] [,3]

[1,] 15 42 77

[2,] 32 72 120

[3,] 55 108 169

[4,] 84 150 224

> #4)Multiplication(by constant):-

> mult<-P\*5

> print(mult)

[,1] [,2] [,3]

[1,] 25 30 35

[2,] 40 45 50

[3,] 55 60 65

[4,] 70 75 80

> #5)Division:-

> div<-P/Q

> div

[,1] [,2] [,3]

[1,] 1.666667 0.8571429 0.6363636

[2,] 2.000000 1.1250000 0.8333333

[3,] 2.200000 1.3333333 1.0000000

[4,] 2.333333 1.5000000 1.1428571

>

**#creation of Arrays:-**

vec1 <-c(1,3,5)

vec2 <-c(10,11,12,13,14,15)

res <- array(c(vec1,vec2),dim=c(3,3,2))

print(res)

**#Naming Of Arrays**

col\_names <- c("Col1","Col2","Col3")

row\_names <- c("Row1","Row2","Row3")

matrix\_names <- c("Matrix1","Matrix2")

res <- array(c(vec1,vec2),dim=c(3,3,2),dimnames=list(row\_names,col\_names,matrix\_names))

print(res)

**OUTPUT: -**

#creation of Arrays:-

> vec1 <-c(1,3,5)

> vec2 <-c(10,11,12,13,14,15)

> res <- array(c(vec1,vec2),dim=c(3,3,2))

> print(res)

, , 1

[,1] [,2] [,3]

[1,] 1 10 13

[2,] 3 11 14

[3,] 5 12 15

, , 2

[,1] [,2] [,3]

[1,] 1 10 13

[2,] 3 11 14

[3,] 5 12 15

>

> #Naming Of Arrays

> col\_names <- c("Col1","Col2","Col3")

> row\_names <- c("Row1","Row2","Row3")

> matrix\_names <- c("Matrix1","Matrix2")

> res <- array(c(vec1,vec2),dim=c(3,3,2),dimnames=list(row\_names,col\_names,matrix\_names))

> print(res)

, , Matrix1

Col1 Col2 Col3

Row1 1 10 13

Row2 3 11 14

Row3 5 12 15

, , Matrix2

Col1 Col2 Col3

Row1 1 10 13

Row2 3 11 14

Row3 5 12 15

**#Creation Of DataFrame**

stud.data<- data.frame(

student\_id = c (1:5),

student\_name = c("Shubham","Arpita","Nishka","Gunjan","Sumit"),

class = c("MBA","MCA","MBA","IMCA","MCA"),

roll\_no=c(20,45,78,12,50)

)

print(stud.data)

**#Operations on DataFrame:-**

**1)Extracting specific columns from a data frame**

final <- data.frame(stud.data$student\_id,stud.data$class)

print(final)

**2)Modification of DataFrame:**

**#Adding row in the data frame**

x <- list(6,"Vaishali","IMCA",15)

rbind(stud.data,x)

**#Adding column in the data frame**

y <- c("Moradabad","Lucknow","Etah","Sambhal","Khurja")

cbind(stud.data,city=y)

**#Delete rows from data frame**

stud.data<-stud.data[-1,]

print(stud.data)

**#Delete column from the data frame**

stud.data$roll\_no<-NULL

print(stud.data)

**OUTPUT:-**

#Creation Of DataFrame

> stud.data<- data.frame(

+ student\_id = c (1:5),

+ student\_name = c("Shubham","Arpita","Nishka","Gunjan","Sumit"),

+ class = c("MBA","MCA","MBA","IMCA","MCA"),

+ roll\_no=c(20,45,78,12,50)

+ )

> print(stud.data)

student\_id student\_name class roll\_no

1 1 Shubham MBA 20

2 2 Arpita MCA 45

3 3 Nishka MBA 78

4 4 Gunjan IMCA 12

5 5 Sumit MCA 50

> #operations on DataFrame:-

> #Extracting specific columns from a data frame

> final <- data.frame(stud.data$student\_id,stud.data$class)

> print(final)

stud.data.student\_id stud.data.class

1 1 MBA

2 2 MCA

3 3 MBA

4 4 IMCA

5 5 MCA

> #Modification of DataFrame

> #Adding row in the data frame

> x <- list(6,"Vaishali","IMCA",15)

> rbind(stud.data,x)

student\_id student\_name class roll\_no

1 1 Shubham MBA 20

2 2 Arpita MCA 45

3 3 Nishka MBA 78

4 4 Gunjan IMCA 12

5 5 Sumit MCA 50

6 6 Vaishali IMCA 15

> #Adding column in the data frame

> y <- c("Moradabad","Lucknow","Etah","Sambhal","Khurja")

> cbind(stud.data,city=y)

student\_id student\_name class roll\_no city

1 1 Shubham MBA 20 Moradabad

2 2 Arpita MCA 45 Lucknow

3 3 Nishka MBA 78 Etah

4 4 Gunjan IMCA 12 Sambhal

5 5 Sumit MCA 50 Khurja

> #Delete rows from data frame

> stud.data<-stud.data[-1,]

> print(stud.data)

student\_id student\_name class roll\_no

2 2 Arpita MCA 45

3 3 Nishka MBA 78

4 4 Gunjan IMCA 12

5 5 Sumit MCA 50

> #Delete column from the data frame

> stud.data$roll\_no<-NULL

> print(stud.data)

student\_id student\_name class

2 2 Arpita MCA

3 3 Nishka MBA

4 4 Gunjan IMCA

5 5 Sumit MCA

>

**#creation of List:-**

list\_1<-list("Shubham","Arpita","Vaishali")

list\_1

list\_data<-list("Shubham","Arpita",c(1,2,3,4,5),TRUE,FALSE,22.5,12L)

print(list\_data)

**#Operation on lists:-**

**1)Giving name to list:-**

list\_data <- list(c("Shubham","Nishka","Gunjan"), matrix(c(40,80,60,70,90,80), nrow = 2),

list("BCA","MCA","B.tech"))

names(list\_data) <- c("Students", "Marks", "Course")

list\_data

**2)Accessing elements using index:-**

print(list\_data[1])

**3)Accessing elements using names:-**

print(list\_data["Students"])

print(list\_data$Marks)

**4)Merging Lists:-**

Even\_list <- list(2,4,6)

Odd\_list <- list(1,3,5)

# Merging the two lists.

merged.list <- list(Even\_list,Odd\_list)

print(merged.list)

--------------------------------------------------------------------------------------------------------------------------

**OUTPUT: -**

#creation of List:-

> list\_1<-list("Shubham","Arpita","Vaishali")

> list\_1

[[1]]

[1] "Shubham"

[[2]]

[1] "Arpita"

[[3]]

[1] "Vaishali"

> list\_data<-list("Shubham","Arpita",c(1,2,3,4,5),TRUE,FALSE,22.5,12L)

> print(list\_data)

[[1]]

[1] "Shubham"

[[2]]

[1] "Arpita"

[[3]]

[1] 1 2 3 4 5

[[4]]

[1] TRUE

[[5]]

[1] FALSE

[[6]]

[1] 22.5

[[7]]

[1] 12

> #Operation on lists:-

> #1)Giving name to list:-

> list\_data <- list(c("Shubham","Nishka","Gunjan"), matrix(c(40,80,60,70,90,80), nrow = 2),

+ list("BCA","MCA","B.tech"))

> names(list\_data) <- c("Students", "Marks", "Course")

> list\_data

$Students

[1] "Shubham" "Nishka" "Gunjan"

$Marks

[,1] [,2] [,3]

[1,] 40 60 90

[2,] 80 70 80

$Course

$Course[[1]]

[1] "BCA"

$Course[[2]]

[1] "MCA"

$Course[[3]]

[1] "B.tech"

> #2)Accessing elements using index:-

> print(list\_data[1])

$Students

[1] "Shubham" "Nishka" "Gunjan"

> #3)Accessing elements using names:-

> print(list\_data["Students"])

$Students

[1] "Shubham" "Nishka" "Gunjan"

> print(list\_data$Marks)

[,1] [,2] [,3]

[1,] 40 60 90

[2,] 80 70 80

> #4)Merging Lists:-

> Even\_list <- list(2,4,6)

> Odd\_list <- list(1,3,5)

> # Merging the two lists.

> merged.list <- list(Even\_list,Odd\_list)

> print(merged.list)

[[1]]

[[1]][[1]]

[1] 2

[[1]][[2]]

[1] 4

[[1]][[3]]

[1] 6

[[2]]

[[2]][[1]]

[1] 1

[[2]][[2]]

[1] 3

[[2]][[3]]

[1] 5

**Experiment No:** 3

**Experiment Name:** Write program to demonstrate Loops & Vectorization of Missing Values.

**Name:**

**Roll No:-**

**#using for loop**

week<-c('Sunday',

'Monday',

'Tuesday',

'Wednesday',

'Thursday',

'Friday',

'saturday')

for(day in week)

{

print(day)

}

**#using while loop**

val=1

while(val<=5)

{

print(val)

val=val+1

}

**#using repeat loop**

val=1

repeat

{

print(val)

val=val+1

if(val>5)

{

break

}

}

**OUTPUT :**

**For-loop**

[1] "Sunday"

[1] "Monday"

[1] "Tuesday"

[1] "Wednesday"

[1] "Thursday"

[1] "Friday"

[1] "saturday"

**While-loop**

[1] 1

[1] 2

[1] 3

[1] 4

[1] 5

**Repeat-loop**

[1] 1

[1] 2

[1] 3

[1] 4

[1] 5

**Experiment No:** 4

**Experiment Name:** Demonstrate Importing and exporting data.

**Name:**

**Roll No:-**

**#IMPORT**

getwd()

#Importing csv file.

path<-"C:/Users/Leena/OneDrive/Documents/candidate-elimination.csv"

content<-read.csv(path)

print(content)

#Importing Text file.

x<-read.table("C:/Users/Leena/OneDrive/Documents/file.txt",header=FALSE)

print(x)

#Importing CSV file using csv2.

x<-read.csv2("C:/Users/Leena/OneDrive/Documents/candidate-elimination.csv")

print(x)

**OUTPUT**

SKY TEMP HUMID WIND WATER FOREST OUTPUT

1 sunny warm normal strong warm same yes

2 sunny warm high strong warm same yes

3 rainy cold high strong warm change no

4 sunny warm high strong cool change yes

SKY.TEMP.HUMID.WIND.WATER.FOREST.OUTPUT

1 sunny,warm,normal,strong,warm,same,yes

2 sunny,warm,high,strong,warm,same,yes

3 rainy,cold,high,strong,warm,change,no

4 sunny,warm,high,strong,cool,change,yes

SKY.TEMP.HUMID.WIND.WATER.FOREST.OUTPUT

1 sunny,warm,normal,strong,warm,same,yes

2 sunny,warm,high,strong,warm,same,yes

3 rainy,cold,high,strong,warm,change,no

4 sunny,warm,high,strong,cool,change,yes

**# EXPORT**

**1).Export a data frame to a text file using write.table().**

df=data.frame(

"Name"=c("Leena","Roshani","Komal"),

"Language"=c("R","Python","Java"),

"Age"=c(22,25,24)

)

write.table(df,

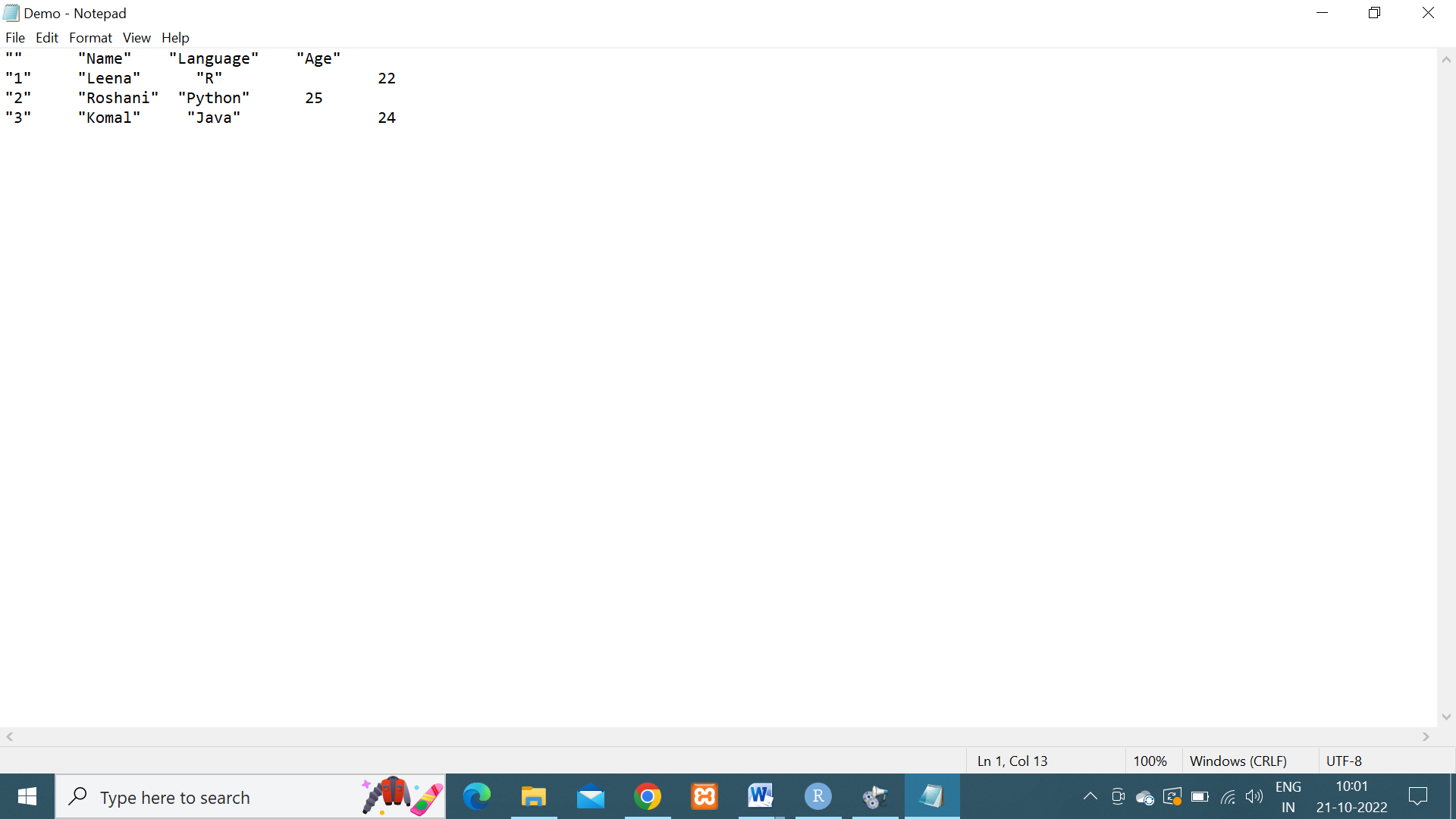
file="Demo.txt",

sep = "\t",

row.names = TRUE,

col.names = NA)

**OUTPUT :**



2).**Exporting Data to a csv file.**

df=data.frame(

"Name"=c("Ankit","Manthan","Pranav"),

"Language"=c("C Programming","Java","HTML"),

"Age"=c(28,27,29)

)

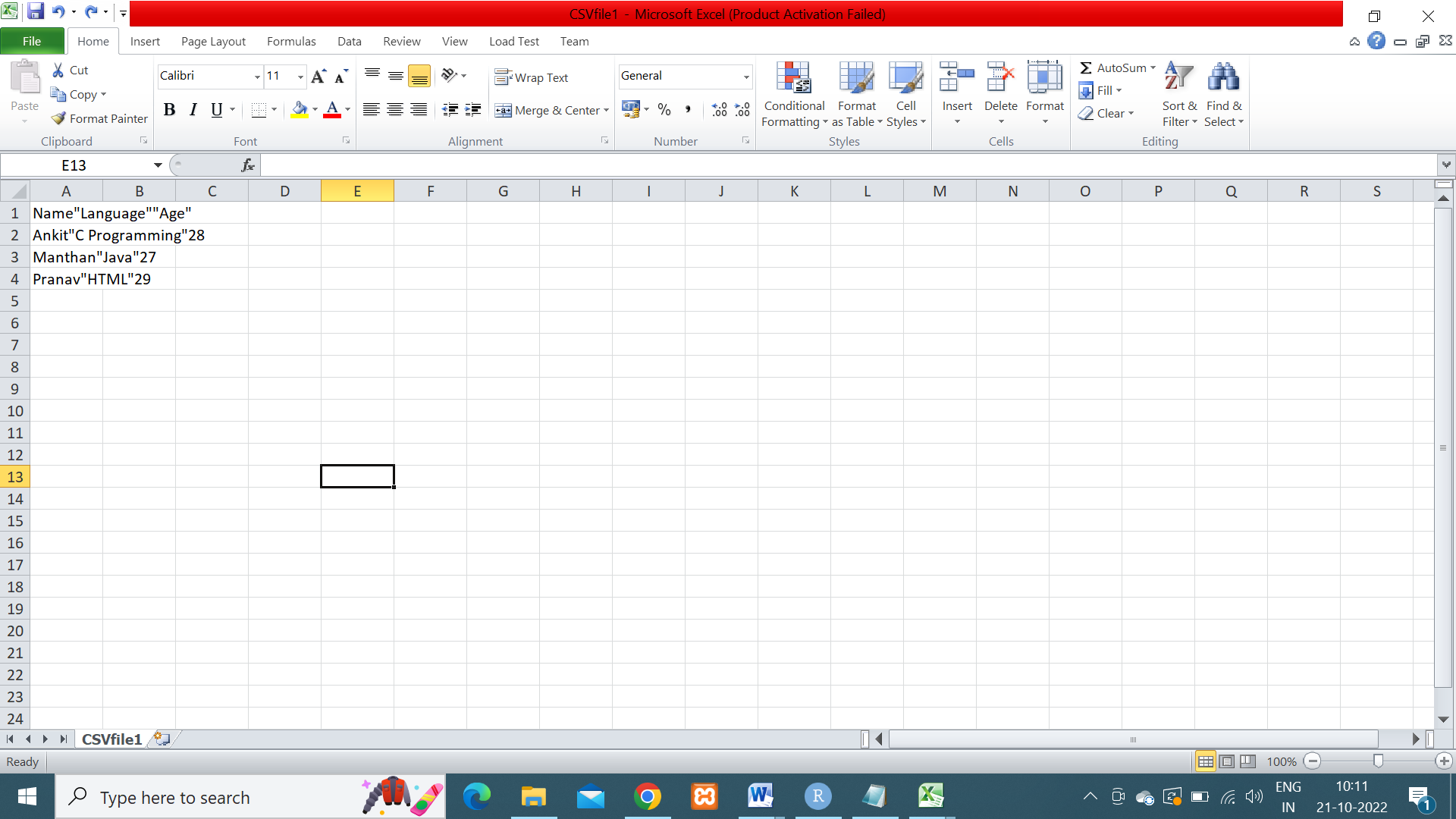
write.table(df,

file="myFile.csv",

sep = "\t",

row.names = FALSE)

**OUTPUT :**



**3)Exporting data to a csv2 file**

library(readr)

df2=data.frame(

"Name"=c("Swati","Anushka","Ashish","Kalpesh"),

"Language"=c("R","Python","Java","PHP"),

"Age"=c(22,25,45,23),

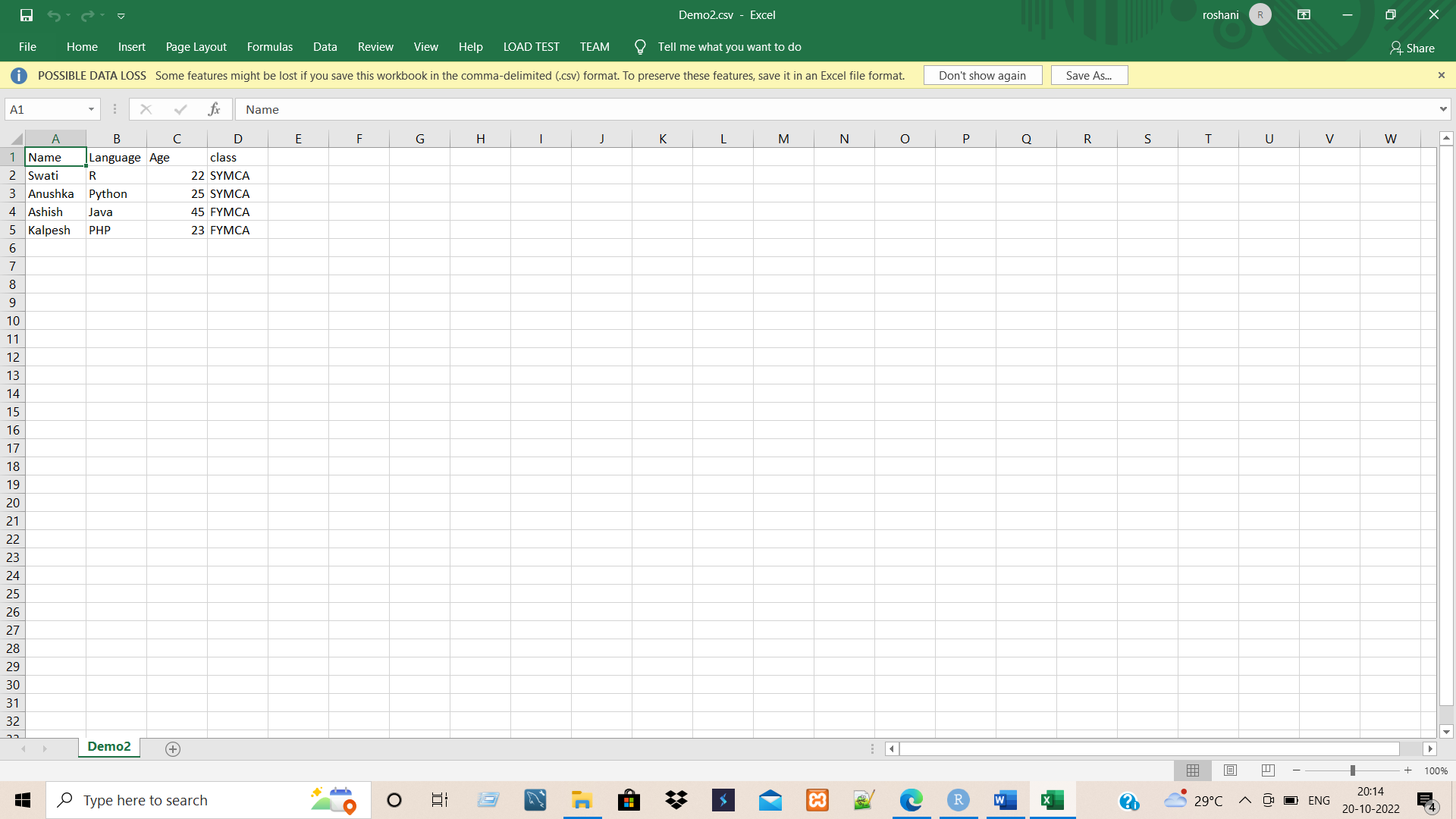
"class"=c("SYMCA","SYMCA","FYMCA","FYMCA")

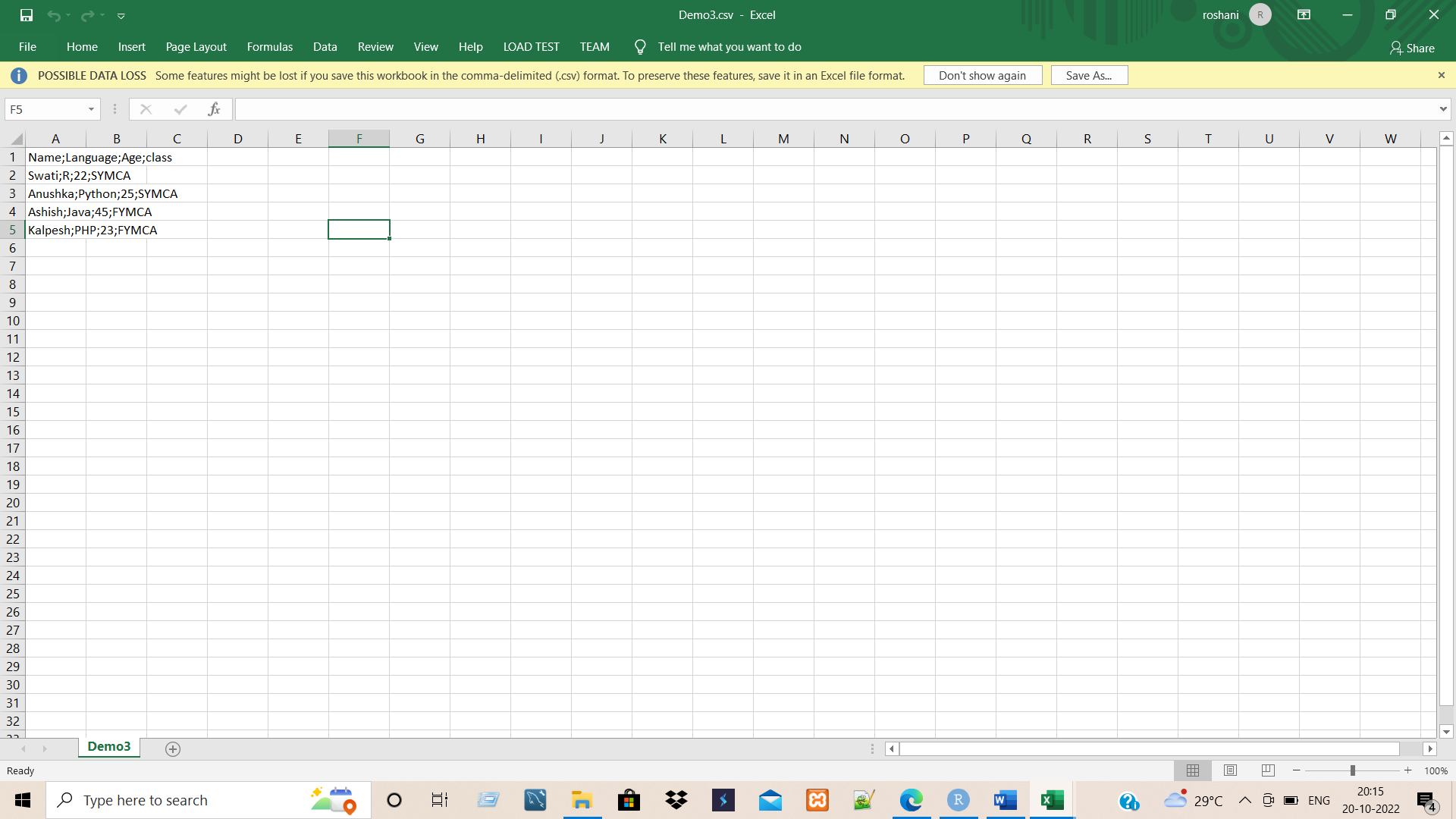
)

write\_csv(df2,path="Demo2.csv")

write\_csv2(df2,path="Demo3.csv")

**OUTPUT**





**4)Exporting data using write\_tsv()function**

getwd()

#library(readr)

df2=data.frame(

"Name"=c("Swati","Anushka","Ashish","Kalpesh"),

"Language"=c("R","Python","Java","PHP"),

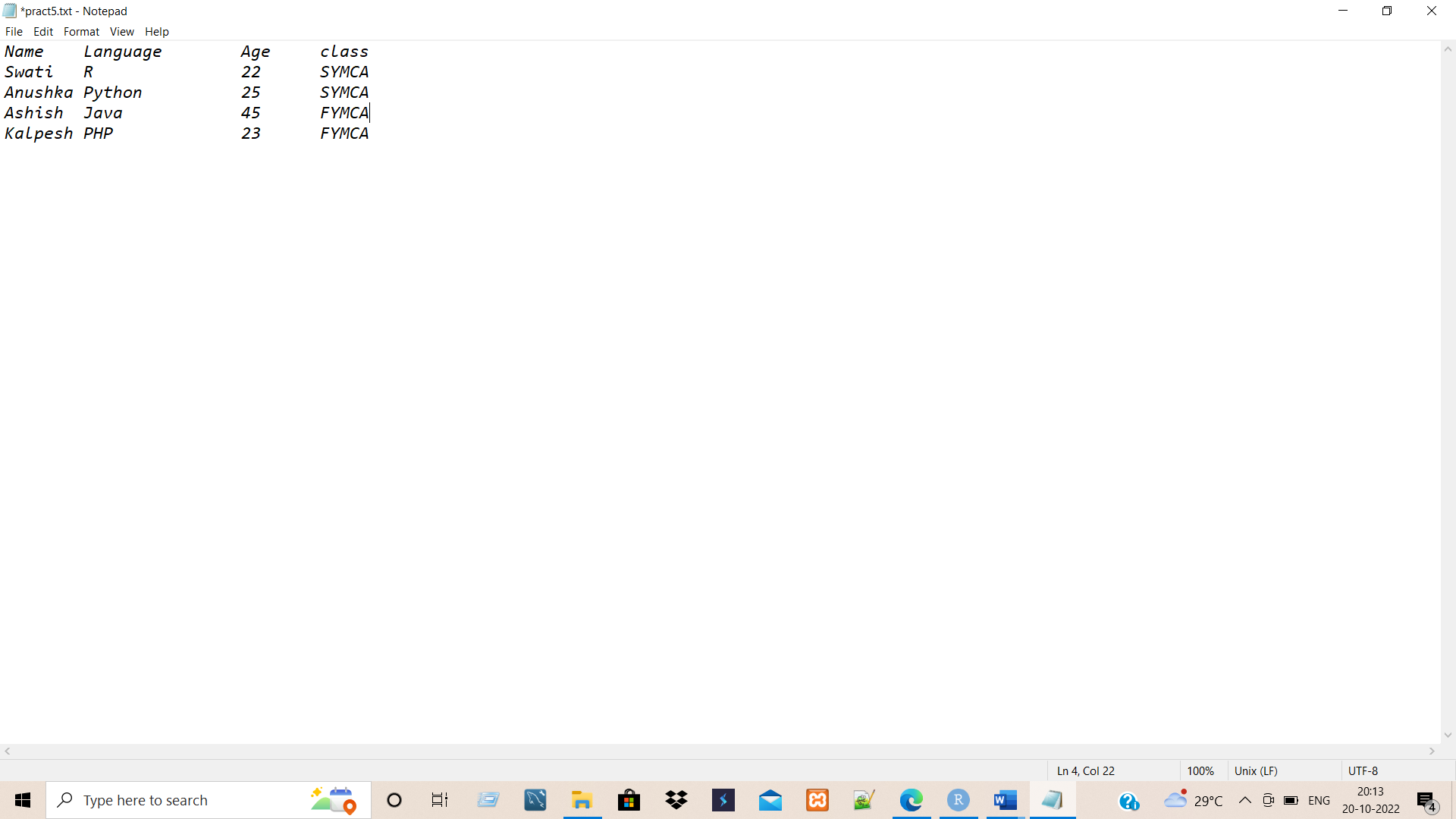
"Age"=c(22,25,45,23),

"class"=c("SYMCA","SYMCA","FYMCA","FYMCA")

)

write\_tsv(df2,path="pract5.txt")

**OUTPUT**



**Experiment No:** 5

**Experiment Name:** Write program for validating and exploring data manipulation (Summarizing, Sorting, Subsetting, Merging, Joining).

**Name:**

**Roll No:-**

**1)Summarizing:-**

#create a data frame

data1<-data.frame(player=c('A','B','c','D','E'),

runs=c(100,200,105,50,90),

wickets=c(15,20,8,5,8)

)

data1

#summarize method

summarize(data1,sum(runs),mean(runs),mode(wickets))

//summarize(data1)

------------------------------------------------------------------------------------------------------------------------------

**2)Sorting:-**

#creating data frame

dataBook=data.frame(Customers=c("Ruhi","James","Heera","Shubham","Joe","Priya"),

Products=c("ProdA","ProdB","ProdC","ProdD","ProdE","prodF"),

Salary=c(500,600,450,700,300,400))

dataBook

#sorting the data frame in ascending order

arrange(dataBook,Salary)

#sorting the data frame in descending order

dataBook%>%arrange(desc(Salary))

------------------------------------------------------------------------------------------------------------------------------

**3)Subsetting:-**

#Subsetting in R using []operator:

#create vector

x<-1:15

cat("Original vector:",x,"\n")

#subsetting vector:

cat("First 5 values of vector:",x[1:5],"\n")

cat("Without values present at index 1,2and 3",x[-c(1,2,3),"\n"])

#Subsetting in R using [[]]operator:

#create list:

ls<-list(a=1,b=2,c=10,d=20)

cat("Original List:\n")

print(ls)

#select first element of list:

cat("Element of list:",ls[[3]],"\n")

#Subsetting using c() function:

ls2<-list(a=list(x=1,y="students"),b=1:10)

ls2

cat("Using c() function:\n")

//print(ls2[[c(1,2)]])

//print(ls2[[1]][[2]])

#Subsetting Using $ operator:

ls3<-list(a="Roshani",b=1,c="Hello")

ls3

cat("Using $ operator:\n")

print(ls3$a)

---------------------------------------------------------------------------------------------------------------

**4)Merging: -**

#Merge DataFrames by Row Names:-

data\_frame1<-data.frame(No=c(1:5),

Name=letters[1:5],

Salary=c(200,200,300,NA,300)

)

data\_frame1

data\_frame2<-data.frame(No=c(6:8),

Name=letters[8:10],

Salary=c(400,350,NA)

)

data\_frame2

data\_frame\_merge<-merge(data\_frame1,data\_frame2,by='row.names',all=TRUE)

print("Merged DataFrame")

print(data\_frame\_merge)

----------------------------------------------------------------------------------------------------------------

**5)Joining:-**

#Using Inner join:-

data1<-data.frame(ID=c(1:5))

data2<-data.frame(ID=c(4:8))

inner\_join(data1,data2,by="ID")

#Using Left join:-

data1<-data.frame(ID=c(1:5),

Name=c("Rutuja","Lokesh","Ram","Purvi","Nita"))

data2<-data.frame(ID=c(4:8),

Marks=c(70,85,80,90,75))

left\_join(data1,data2,by="ID")

------------------------------------------------------------------------------------------------------------------------------------

**OUTPUT: -**

#1)Summarizing:-

> #create a data frame

> data1<-data.frame(player=c('A','B','c','D','E'),

+ runs=c(100,200,105,50,90),

+ wickets=c(15,20,8,5,8)

+ )

> data1

player runs wickets

1 A 100 15

2 B 200 20

3 c 105 8

4 D 50 5

5 E 90 8

> #summarize method

> summarize(data1,sum(runs),mean(runs),mode(wickets))

sum(runs) mean(runs) mode(wickets)

1 545 109 numeric

> #-------------------------------------------------------------------------------

> #2)Sorting:-

> #creating data frame

> dataBook=data.frame(Customers=c("Ruhi","James","Heera","Shubham","Joe","Priya"),

+ Products=c("ProdA","ProdB","ProdC","ProdD","ProdE","prodF"),

+ Salary=c(500,600,450,700,300,400))

> dataBook

Customers Products Salary

1 Ruhi ProdA 500

2 James ProdB 600

3 Heera ProdC 450

4 Shubham ProdD 700

5 Joe ProdE 300

6 Priya prodF 400

> #sorting the data frame in ascending order

> arrange(dataBook,Salary)

Customers Products Salary

1 Joe ProdE 300

2 Priya prodF 400

3 Heera ProdC 450

4 Ruhi ProdA 500

5 James ProdB 600

6 Shubham ProdD 700

> #sorting the data frame in descending order

> dataBook%>%arrange(desc(Salary))

Customers Products Salary

1 Shubham ProdD 700

2 James ProdB 600

3 Ruhi ProdA 500

4 Heera ProdC 450

5 Priya prodF 400

6 Joe ProdE 300

> #-------------------------------------------------------------------------------

> #3)Subsetting:-

> #Subsetting in R using []operator:

> #create vector

> x<-1:15

> cat("Original vector:",x,"\n")

Original vector: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

> #subsetting vector:

> cat("First 5 values of vector:",x[1:5],"\n")

First 5 values of vector: 1 2 3 4 5

> cat("Without values present at index 1,2and 3:",x[-c(1,2,3)])

Without values present at index 1,2and 3: 4 5 6 7 8 9 10 11 12 13 14 15> #Subsetting in R using [[]]operator:

> #create list:

> ls<-list(a=1,b=2,c=10,d=20)

> cat("Original List:\n")

Original List:

> print(ls)

$a

[1] 1

$b

[1] 2

$c

[1] 10

$d

[1] 20

> #select first element of list:

> cat("Element of list:",ls[[3]],"\n")

Element of list: 10

> #Subsetting using c() function:

> ls2<-list(a=list(x=1,y="students"),b=1:10)

> ls2

$a

$a$x

[1] 1

$a$y

[1] "students"

$b

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

> cat("Using c() function:\n")

Using c() function:

> print(ls2[[c(1,2)]])

[1] "students"

> print(ls2[[1]][[2]])

[1] "students"

> #Subsetting Using $ operator:

> ls3<-list(a="Roshani",b=1,c="Hello")

> ls3

$a

[1] "Roshani"

$b

[1] 1

$c

[1] "Hello"

> cat("Using $ operator:\n")

Using $ operator:

> print(ls3$a)

[1] "Roshani"

> #----------------------------------------------------------------------------------

> #4)Merging:-

> #Merge DataFrames by Row Names:-

> data\_frame1<-data.frame(No=c(1:5),

+ Name=letters[1:5],

+ Salary=c(200,200,300,NA,300)

+ )

> data\_frame1

No Name Salary

1 1 a 200

2 2 b 200

3 3 c 300

4 4 d NA

5 5 e 300

>

> data\_frame2<-data.frame(No=c(6:8),

+ Name=letters[8:10],

+ Salary=c(400,350,NA)

+ )

> data\_frame2

No Name Salary

1 6 h 400

2 7 i 350

3 8 j NA

>

> data\_frame\_merge<-merge(data\_frame1,data\_frame2,by='row.names',all=TRUE)

> print("Merged DataFrame")

[1] "Merged DataFrame"

> print(data\_frame\_merge)

Row.names No.x Name.x Salary.x No.y Name.y Salary.y

1 1 1 a 200 6 h 400

2 2 2 b 200 7 i 350

3 3 3 c 300 8 j NA

4 4 4 d NA NA <NA> NA

5 5 5 e 300 NA <NA> NA

> #-------------------------------------------------------------------------------

> #5)Joining:-

> #Using Inner join:-

> data1<-data.frame(ID=c(1:5))

> data2<-data.frame(ID=c(4:8))

> inner\_join(data1,data2,by="ID")

ID

1 4

2 5

>

> #Using Left join:-

> data1<-data.frame(ID=c(1:5),

+ Name=c("Rutuja","Lokesh","Ram","Purvi","Nita"))

> data2<-data.frame(ID=c(4:8),

+ Marks=c(70,85,80,90,75))

> left\_join(data1,data2,by="ID")

ID Name Marks

1 1 Rutuja NA

2 2 Lokesh NA

3 3 Ram NA

4 4 Purvi 70

5 5 Nita 85

**#Validating data:-**

data(cars)

head(cars, 3)

library(validate)

rules <- validator(speed >= 0,

dist >= 0,

speed/dist <= 1.5,

cor(speed, dist)>=0.2)

out <- confront(cars, rules)

summary(out)

**Output: -**

data(cars)

> head(cars, 3)

speed dist

1 4 2

2 4 10

3 7 4

>

> library(validate)

> rules <- validator(speed >= 0,

+ dist >= 0,

+ speed/dist <= 1.5,

+ cor(speed, dist)>=0.2)

> out <- confront(cars, rules)

> summary(out)

name items passes fails nNA error warning expression

1 V1 50 50 0 0 FALSE FALSE speed - 0 >= -1e-08

2 V2 50 50 0 0 FALSE FALSE dist - 0 >= -1e-08

3 V3 50 48 2 0 FALSE FALSE speed/dist <= 1.5

4 V4 1 1 0 0 FALSE FALSE cor(speed, dist) >= 0.2

**Experiment No:** 6

**Experiment Name:** Write program to implement the following analysis techniques using R.

1.Statistical hypothesis generation and testing

2. Chi-Square test

3. t-Test

4. Correlation analysis

**Name:**

**Roll No:-**

**1)Stastical hypothesis testing:-**

#One-sample T-testing:

x<-rnorm(100)#sample vector

t.test(x,mu=5)#one-sample t-test

#two-sample T-testing:

x<-rnorm(100)

y<-rnorm(100)

t.test(x,y)

#Directional Hypothesis:-

t.test(x,mu=2,alternative = 'greater')

#one sample u-test:-

wilcox.test(y,exact = FALSE)

#Two sample u-test:-

wilcox.test(x,y)

**2)Correlation Test:-**

cor.test(matcars$mpg,matcars$hp)

**3)Chi-Square Test:-**

library(MASS)

#create DataFrame:

print(str(survey))

# Create a data frame from the main data set.

stu\_data = data.frame(survey$Smoke,survey$Exer)

# Create a contingency table with the needed variables.

stu\_data = table(survey$Smoke,survey$Exer)

print(stu\_data)

**OUTPUT:-**

#1)Stastical hypothesis testing:-

> #One-sample T-testing:

> x<-rnorm(100)#sample vector

> t.test(x,mu=5)#one-sample t-test

One Sample t-test

data: x

t = -52.314, df = 99, p-value < 2.2e-16

alternative hypothesis: true mean is not equal to 5

95 percent confidence interval:

-0.2298852 0.1523448

sample estimates:

mean of x

-0.03877023

> #two-sample T-testing:

> x<-rnorm(100)

> y<-rnorm(100)

> t.test(x,y)

Welch Two Sample t-test

data: x and y

t = -0.062003, df = 197.96, p-value = 0.9506

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

-0.2842159 0.2668885

sample estimates:

mean of x mean of y

0.04941380 0.05807748

> #Directional Hypothesis:-

> t.test(x,mu=2,alternative = 'greater')

One Sample t-test

data: x

t = -19.884, df = 99, p-value = 1

alternative hypothesis: true mean is greater than 2

95 percent confidence interval:

-0.1134708 Inf

sample estimates:

mean of x

0.0494138

> #one sample u-test:-

> wilcox.test(y,exact = FALSE)

Wilcoxon signed rank test with continuity correction

data: y

V = 2589, p-value = 0.8272

alternative hypothesis: true location is not equal to 0

> #Two sample u-test:-

> wilcox.test(x,y)

Wilcoxon rank sum test with continuity correction

data: x and y

W = 5039, p-value = 0.9251

alternative hypothesis: true location shift is not equal to 0

> #2)Correlation Test:-

> cor.test(mtcars$mpg,mtcars$hp)

Pearson's product-moment correlation

data: mtcars$mpg and mtcars$hp

t = -6.7424, df = 30, p-value = 1.788e-07

alternative hypothesis: true correlation is not equal to 0

95 percent confidence interval:

-0.8852686 -0.5860994

sample estimates:

cor

-0.7761684

> #3)Chi-Square Test:-

> library(MASS)

> #create DataFrame:

> print(str(survey))

'data.frame': 237 obs. of 12 variables:

$ Sex : Factor w/ 2 levels "Female","Male": 1 2 2 2 2 1 2 1 2 2 ...

$ Wr.Hnd: num 18.5 19.5 18 18.8 20 18 17.7 17 20 18.5 ...

$ NW.Hnd: num 18 20.5 13.3 18.9 20 17.7 17.7 17.3 19.5 18.5 ...

$ W.Hnd : Factor w/ 2 levels "Left","Right": 2 1 2 2 2 2 2 2 2 2 ...

$ Fold : Factor w/ 3 levels "L on R","Neither",..: 3 3 1 3 2 1 1 3 3 3 ...

$ Pulse : int 92 104 87 NA 35 64 83 74 72 90 ...

$ Clap : Factor w/ 3 levels "Left","Neither",..: 1 1 2 2 3 3 3 3 3 3 ...

$ Exer : Factor w/ 3 levels "Freq","None",..: 3 2 2 2 3 3 1 1 3 3 ...

$ Smoke : Factor w/ 4 levels "Heavy","Never",..: 2 4 3 2 2 2 2 2 2 2 ...

$ Height: num 173 178 NA 160 165 ...

$ M.I : Factor w/ 2 levels "Imperial","Metric": 2 1 NA 2 2 1 1 2 2 2 ...

$ Age : num 18.2 17.6 16.9 20.3 23.7 ...

NULL

> # Create a data frame from the main data set.

> stu\_data = data.frame(survey$Smoke,survey$Exer)

> # Create a contingency table with the needed variables.

> stu\_data = table(survey$Smoke,survey$Exer)

> print(stu\_data)

Freq None Some

Heavy 7 1 3

Never 87 18 84

Occas 12 3 4

Regul 9 1 7

**Experiment No:** 7

**Experiment Name:** Write program to implement the following analysis techniques using R.

4. Analysis of variance (ANOVA)

**Name:**

**Roll No:-**

**Analysis of variance test**

ANOVA also known as Analysis of variance is used to investigate relations between categorical variables and continuous variable in R Programming. It is a type of hypothesis testing for population variance.

**R – ANOVA Test**

ANOVA test involves setting up:

* **Null Hypothesis:** All population means are equal.
* **Alternate Hypothesis:** At least one population mean is different from other.

ANOVA tests are of two types:

* **One-way ANOVA:** It takes one categorical group into consideration.
* **Two-way ANOVA:** It takes two categorical group into consideration.

**The Dataset we used for Analysis of Variance test**

The mtcars (motor trend car road test) dataset is used which consist of 32 car brands and 11 attributes. The dataset comes preinstalled in **dplyr** package in R.

To get started with ANOVA, we need to install and load the **dplyr** package.

**Performing One Way ANOVA test in R language**

One-way ANOVA test is performed using mtcars dataset which comes preinstalled with dplyr package between disp attribute, a continuous attribute and gear attribute, a categorical attribute.

**Program: -**

**# Installing the package**

install.packages("dplyr")

**# Loading the package**

library(dplyr)

**# Variance in mean within group and between group**

boxplot(mtcars$disp~factor(mtcars$gear) xlab = "gear", ylab = "disp")

**# Step 1: Setup Null Hypothesis and Alternate Hypothesis**

**# H0 = mu = mu01 = mu02(There is no difference**

**# between average displacement for different gear)**

**# H1 = Not all means are equal**

**# Step 2: Calculate test statistics using aov function**

mtcars\_aov <- aov(mtcars$disp~factor(mtcars$gear))

summary(mtcars\_aov)

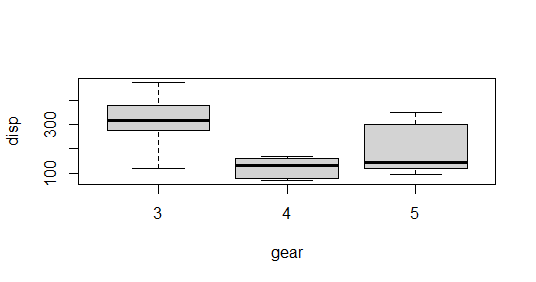
**# Step 3: Calculate F-Critical Value**

**# For 0.05 Significant value, critical value = alpha = 0.05**

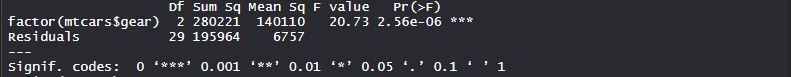
**# Step 4: Compare test statistics with F-Critical value**

**# and conclude test p < alpha, Reject Null Hypothesis**

**Output:**



The box plot shows the mean values of gear with respect of displacement. Hear categorical variable is gear on which factor function is used and continuous variable is disp.



The summary shows that the gear attribute is very significant to displacement (Three stars denoting it). Also, the P value is less than 0.05, so proves that gear is significant to displacement i.e related to each other and we reject the Null Hypothesis.

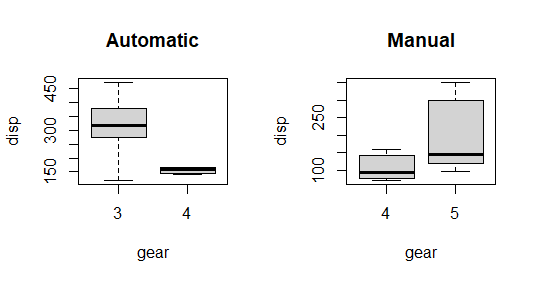
**Performing Two Way ANOVA test in R**

Two-way ANOVA test is performed using mtcars dataset which comes preinstalled with dplyr package between disp attribute, a continuous attribute and gear attribute, a categorical attribute, am attribute, a categorical attribute.

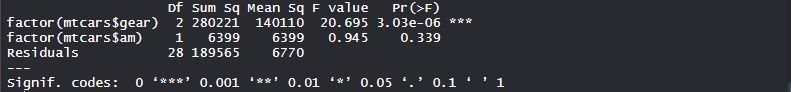
**Program: -**

|  |
| --- |
| **# Installing the package**  install.packages("dplyr")    **# Loading the package**  library(dplyr)    **# Variance in mean within group and between group**  boxplot(mtcars$disp~mtcars$gear, subset = (mtcars$am == 0),          xlab = "gear", ylab = "disp", main = "Automatic")  boxplot(mtcars$disp~mtcars$gear, subset = (mtcars$am == 1),              xlab = "gear", ylab = "disp", main = "Manual")    **# Step 1: Setup Null Hypothesis and Alternate Hypothesis**  **# H0 = mu0 = mu01 = mu02(There is no difference between**  **# average displacement for different gear)**  **# H1 = Not all means are equal**    **# Step 2: Calculate test statistics using aov function**  mtcars\_aov2 <- aov(mtcars$disp~factor(mtcars$gear) \*factor(mtcars$am))  summary(mtcars\_aov2)    **# Step 3: Calculate F-Critical Value**  **# For 0.05 Significant value, critical value = alpha = 0.05**  **# Step 4: Compare test statistics with F-Critical value**  **# and conclude test p < alpha, Reject Null Hypothesis** |

**Output:**



The box plot shows the mean values of gear with respect to displacement. Hear categorical variables are gear and am on which factor function is used and continuous variable is disp.



The summary shows that the gear attribute is very significant to displacement (Three stars denoting it) and am attribute is not much significant to displacement. P-value of gear is less than 0.05, so it proves that gear is significant to displacement i.e related to each other. P-value of am is greater than 0.05, am is not significant to displacement i.e not related to each other.

**Results**

We see significant results from boxplots and summaries.

* Displacement is strongly related to Gears in cars i.e displacement is dependent on gears with p < 0.05.
* Displacement is strongly related to Gears but not related to transmission mode in cars with p 0.05 with am.

**Experiment No:** 8

**Experiment Name:** Write program to implement the following analysis techniques using R.

Regression analysis

**Name:**

**Roll No:-**

**Regression analysis test**

Regression analysis is a statistical tool to estimate the relationship between two or more variables. There is always one response variable and one or more predictor variables. Regression analysis is widely used to fit the data accordingly and further, predicting the data for forecasting. It helps businesses and organizations to learn about the behavior of their product in the market using the dependent/response variable and independent/predictor variable.

**Types of Regression in R**

There are mainly three types of Regression in R programming that is widely used. They are:

* [Linear Regression](https://www.geeksforgeeks.org/simple-linear-regression-using-r/)
* [Multiple Regression](https://www.geeksforgeeks.org/multiple-linear-regression-using-r/)
* [Logistic Regression](https://www.geeksforgeeks.org/logistic-regression-in-r-programming/)

**Linear Regression**

The Linear Regression model is one of the widely used among three of the regression types. In linear regression, the relationship is estimated between two variables i.e., one response variable and one predictor variable. Linear regression produces a straight line on the graph. Mathematically

***where,***

* ***x****indicates predictor or independent variable*
* ***y****indicates response or dependent variable*
* ***a****and****b****are coefficients*

**Implementation in R**

In R programming, [**lm()**](https://www.geeksforgeeks.org/perform-linear-regression-analysis-in-r-programming-lm-function/) function is used to create linear regression model.

***Syntax:****lm(formula)****Parameter:******formula:****represents the formula on which data has to be fitted To know about more optional parameters, use below command in console: help(“lm”)*

**Example:** In this example, let us plot the linear regression line on the graph and predict the weight-based using height.

**Program: -**

|  |
| --- |
| **# R program to illustrate**  **# Linear Regression**  **# Height vector**  x <- c(153, 169, 140, 186, 128,         136, 178, 163, 152, 133)    **# Weight vector**  y <- c(64, 81, 58, 91, 47, 57,         75, 72, 62, 49)    **# Create a linear regression model**  model <- lm(y~x)    **# Print regression model**  print(model)    **# Find the weight of a person With height 182**  df <- data.frame(x = 182)  res <-  predict(model, df)  cat("\nPredicted value of a person                 with height = 182")  print(res)    **# Output to be present as PNG file**  png(file = "linearRegGFG.png")    **# Plot**  plot(x, y, main = "Height vs Weight Regression model")  abline(lm(y~x))    **# Save the file.**  dev.off() |

**Output:**

Call:

lm(formula = y ~ x)

Coefficients:

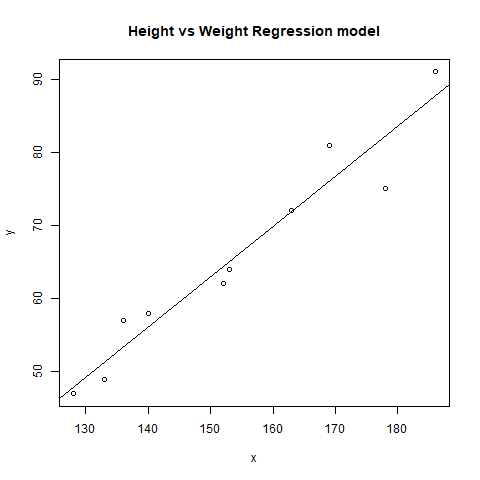
(Intercept) x

-39.7137 0.6847

Predicted value of a person with height = 182

1

84.9098



**Multiple Regression**

Multiple regression is another type of regression analysis technique that is an extension of the linear regression model as it uses more than one predictor variables to create the model. Mathematically,   
 **Implementation in R**

Multiple regression in R programming uses the same **lm()** function to create the model.

***Syntax:****lm(formula, data)*

***Parameters:***

* ***formula:****represents the formula on which data has to be fitted*
* ***data:****represents dataframe on which formula has to be applied*

**Example:** Let us create a multiple regression model of air quality dataset present in R base package and plot the model on the graph.

**Program: -**

|  |
| --- |
| **# R program to illustrate**  **# Multiple Linear Regression**  **# Using airquality dataset**  input <- airquality[1:50,c("Ozone", "Wind", "Temp")]    **# Create regression model**  model <- lm(Ozone~Wind + Temp,data = input)    **# Print the regression model**  cat("Regression model:\n")  print(model)    **# Output to be present as PNG file**  png(file = "multipleRegGFG.png")    **# Plot**  plot(model)    **# Save the file.**  dev.off() |

**Output:**

Regression model:

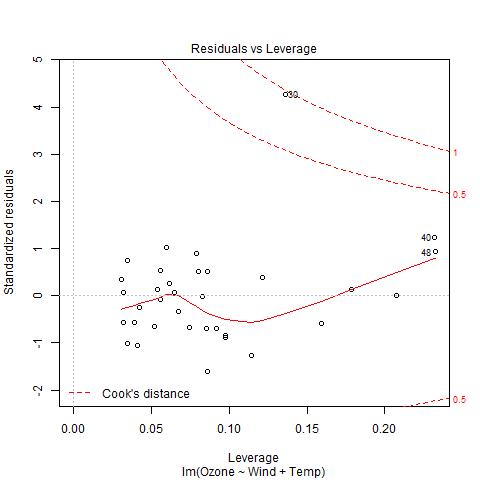
Call:

lm(formula = Ozone ~ Wind + Temp, data = input)

Coefficients:

(Intercept) Wind Temp

-58.239 -0.739 1.329



**Logistic Regression**

Logistic Regression is another widely used regression analysis technique and predicts the value with a range. Moreover, it is used for predicting the values for categorical data. For example, Email is either spam or non-spam, winner or loser, male or female, etc. Mathematically,

***where,***

* ***y****represents response variable*
* ***z****represents equation of independent variables or features*

**Implementation in R**

In R programming, **[glm()](https://www.geeksforgeeks.org/fitting-linear-models-to-the-data-set-in-r-programming-glm-function/)** function is used to create a logistic regression model.

***Syntax:****glm(formula, data, family)*

***Parameters:***

* ***formula:****represents a formula on the basis of which model has to be fitted*
* ***data:****represents dataframe on which formula has to be applied*
* ***family:****represents the type of function to be used. “binomial” for logistic regression*

**Example:**

|  |
| --- |
| **# R program to illustrate**  **# Logistic Regression**  **# Using mtcars dataset**  **# To create the logistic model**  model <- glm(formula = vs ~ wt,family = binomial,data = mtcars)    **# Creating a range of wt values**  x <- seq(min(mtcars$wt),max(mtcars$wt),0.01)    **# Predict using weight**  y <- predict(model, list(wt = x), type = "response")    **# Print model**  print(model)    **# Output to be present as PNG file**  png(file = "LogRegGFG.png")    **# Plot**  plot(mtcars$wt, mtcars$vs, pch = 16, xlab = "Weight", ylab = "VS")  lines(x, y)    **# Saving the file**  dev.off() |

**Output:**

Call: glm(formula = vs ~ wt, family = binomial, data = mtcars)

Coefficients:

(Intercept) wt

5.715 -1.911

Degrees of Freedom: 31 Total (i.e. Null); 30 Residual

Null Deviance: 43.86

Residual Deviance: 31.37 AIC: 35.37

