**R PROGRAMMING**

\* USE ctrl+l TO CLEAR SCREEN.

Declaring variable in R

To declare variable, we need to assign a variable name. The name should not have space. We can use \_ to connect to words.

Eg:

> x<-5 [ <- works as “equals to”(=)]

> x

O/P=> [1] 5

> y<-10

> y

O/P=> [1] 10

> x+y

O/P=> [1] 15

> a=x+y

> a

O/P=> [1] 15

> b<-5

> c<-6

> d<-7

> e<-8

> ls()

O/P=> [1] "a" "b" "c" "d" "e" "x" "y"

ls()=> To display list of variables

rm(list=ls()) => Removes all variables currently loaded in memory in R.

> a<-5

> a

O/P=> [1] 5

> x<-10

> y<-15

> x-y

O/P=> [1] -5

> x+y

O/P=> [1] 25

> x/y

O/P=> [1] 0.6666667

> x\*y

O/P=> [1] 150

> x%%y

O/P=> [1] 10

**VECTOR**

Vector is one dimensional array. We can create a vector with all the basic data type we learnt before. The simplest way to build a vector in R , is to use c command.

> vec\_num<-c(1,10,49)

> vec\_num

O/P=> [1] 1 10 49

> vec\_chr<-c("a","b","c")

> vec\_chr

O/P=> [1] "a" "b" "c"

Adding two vector

> vect\_1<-c(5,6,7)

> vect\_2<-c(8,9,1)

> sum\_vect<-vect\_1+vect\_2

> sum\_vect

O/P=> [1] 13 15 8

> sub\_vect<-vect\_1-vect\_2

> sub\_vect

O/P=> [1] -3 -3 6

> pro\_vect<-vect\_1\*vect\_2

> pro\_vect

O/P=> [1] 40 54 7

> c(1:10)

O/P=> [1] 1 2 3 4 5 6 7 8 9 10

Index starts from 1.

> x1<-c(1,2,3,4,5,6,7,8,9,10)

> x1[1:5]

[1] 1 2 3 4 5

> x1[4]

[1] 4

here 4 is indicating index position.

> a<-c(2,4,6)

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

> sd(a)

[1] 2

> var(a)

[1] 4

> cor(a,b)

[1] 1

**Matrix**

A matrix is a 2 dimensional array.

Syntax:

matrix(data,nrow,ncol,byrow=TRUE)

eg;

byrow=TRUE => horizontal value

> matrix\_a<-matrix(1:10,byrow=TRUE,nrow=5)

> matrix\_a

[,1] [,2]

[1,] 1 2

[2,] 3 4

[3,] 5 6

[4,] 7 8

[5,] 9 10

dim(matrix\_a)=> Display the dimensional of matrix.

> dim(matrix\_a)

[1] 5 3

byrow=FALSE =>Vertical value

> matrix\_b<-matrix(1:10,byrow=FALSE,nrow=5)

> matrix\_b

[,1] [,2]

[1,] 1 6

[2,] 2 7

[3,] 3 8

[4,] 4 9

[5,] 5 10

> matrix\_c<-matrix(1:12,byrow=FALSE,ncol=3,nrow=3)

> matrix\_c

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

[1,] 1 4 7

[2,] 2 5 8

[3,] 3 6 9

Adding one column in matrix

cbind()=> concatenate colums

rbind()=> appends rows

> matrix\_a<-cbind(matrix\_a,c(1:5))

> matrix\_a

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

[1,] 1 2 3 1

[2,] 4 5 6 2

[3,] 7 8 9 3

[4,] 10 11 12 4

[5,] 13 14 15 5

Note: The number of rows of matrices should be equal for cbind work.

> matrix\_a[1,2] => selects the elements at the first row and second column

O/P

[1] 2

> matrix\_a[1:3,2:3] => results in a matrix with data on row 1,2,3 and columns 2,3.

o/p

[,1] [,2]

[1,] 2 3

[2,] 5 6

[3,] 8 9

> matrix\_a[,1] => selects all the elements of first column.

[1] 1 4 7 10 13

> matrix\_a[1,] => selects all the elements of first row.

[1] 1 2 3 1

**DATA FRAMES**

How to create the data frames?

We can create a data frame by passing the variable a,b,c,d into the data.frame() function. We can name the columns with name() and simply specify the name of the variables.

Arguments:

df: It can be matrix to convert as a data frame or a collection of variables to join.

stringsAsFactors: Convert string to factor by default

Create a,b,c,d variable

> a1<-c(10,20,30,40)

> b1<-c('book','pen','textbook','pencile\_case')

> c1<-c(TRUE,FALSE,TRUE,FALSE)

> d1<-c(2.5,8,10,7)

> df<-data.frame(a1,b1,c1,d1)

> df

a1 b1 c1 d1

1 10 book TRUE 2.5

2 20 pen FALSE 8.0

3 30 textbook TRUE 10.0

4 40 pencile\_case FALSE 7.0

str(df) => Print the structure(It tells about the data types)

'data.frame': 4 obs. of 4 variables:

$ a1: num 10 20 30 40

$ b1: Factor w/ 4 levels "book","pen","pencile\_case",..: 1 2 4 3

$ c1: logi TRUE FALSE TRUE FALSE

$ d1: num 2.5 8 10 7

## Slice Data Frame

## Select row 1 in column 2

> df[1,2]

[1] book

Levels: book pen pencile\_case textbook

> df[1:2,]

a1 b1 c1 d1

1 10 book TRUE 2.5

2 20 pen FALSE 8.0

> df[,1]

[1] 10 20 30 40

> df[1:3,3:4]

c1 d1

1 TRUE 2.5

2 FALSE 8.0

3 TRUE 10.0

#Name the data frame

> names(df)<-c('ID','items','store','price')

> df

ID items store price

1 10 book TRUE 2.5

2 20 pen FALSE 8.0

3 30 textbook TRUE 10.0

4 40 pencile\_case FALSE 7.0

Retriving values:

> df[,c('ID','store')]

ID store

1 10 TRUE

2 20 FALSE

3 30 TRUE

4 40 FALSE

> df[,c('items','price')]

items price

1 book 2.5

2 pen 8.0

3 textbook 10.0

4 pencile\_case 7.0

> df[1,c('items','price')] => Only value of row 1 will come.

items price

1 book 2.5

> df[1:3,2]

[1] book pen textbook

Levels: book pen pencile\_case textbook

> df[1:3,2:4]

items store price

1 book TRUE 2.5

2 pen FALSE 8.0

3 textbook TRUE 10.0

#Append a column to data frame # use the symbol $ to append

quantity<-c(10,35,45,5)

df$quantity<-quantity

> df

ID items store price quantity

1 10 book TRUE 2.5 10

2 20 pen FALSE 8.0 35

3 30 textbook TRUE 10.0 45

4 40 pencile\_case FALSE 7.0 5

> subset(df,subset=price>5) => record having price greater than 5.

ID items store price quantity

2 20 pen FALSE 8 35

3 30 textbook TRUE 10 45

4 40 pencile\_case FALSE 7 5

> subset(df,subset=ID>20)

ID items store price quantity

3 30 textbook TRUE 10 45

4 40 pencile\_case FALSE 7 5

> df$ID<-NULL => removing the column

> df

items store price quantity

1 book TRUE 2.5 10

2 pen FALSE 8.0 35

3 textbook TRUE 10.0 45

4 pencile\_case FALSE 7.0 5

> df[2,4]<-22 => changing the value

> df

items store price quantity

1 book TRUE 2.5 10

2 pen FALSE 8.0 22

3 textbook TRUE 10.0 45

4 pencile\_case FALSE 7.0 5

order is used for displaying in some order

\*\*\* Its is displaying the price in ascending order

> df1<-df[order(df$price),]

> df1

items store price quantity

1 book TRUE 2.5 10

4 pencile\_case FALSE 7.0 5

2 pen FALSE 8.0 22

3 textbook TRUE 10.0 45

\*\*\*\* Displaying price in descendind order

> df1<-df[order(-df$price),]

> df1

items store price quantity

3 textbook TRUE 10.0 45

2 pen FALSE 8.0 22

4 pencile\_case FALSE 7.0 5

1 book TRUE 2.5 10

> df1<-df[order(-df$price,df$quantity),]

> df1

items store price quantity

3 textbook TRUE 10.0 45

2 pen FALSE 8.0 22

4 pencile\_case FALSE 7.0 5

1 book TRUE 2.5 10

> df1<-df[order(-df$quantity,df$price),]

> head(df1)

items store price quantity

3 textbook TRUE 10.0 45

2 pen FALSE 8.0 22

1 book TRUE 2.5 10

4 pencile\_case FALSE 7.0 5

1. create data frame "student"

prn,name,course,marks.

> prn<-c(01,02,03,04)

> name<-c("Pratikshya","Pratik","Vivan","Pari")

> course<-c("Maths","English","Science","social")

> marks<-c(95,85,72,98)

> student<-data.frame(prn,name,course,marks)

> student

prn name course marks

1 1 Pratikshya Maths 95

2 2 Pratik English 85

3 3 Vivan Science 72

4 4 Pari social 98

2. Slice 2 rows,with PRN and name

> student[1:2,c('prn','name')]

prn name

1 1 Pratikshya

2 2 Pratik

3. Display records with marks >90

> subset(student,subset=marks>90)

prn name course marks

1 1 Pratikshya Maths 95

4 4 Pari social 98

4. Arrange all record in descending order based on "Marks".

> student1<-student[order(-student$marks),]

> student1

prn name course marks

4 4 Pari social 98

1 1 Pratikshya Maths 95

2 2 Pratik English 85

3 3 Vivan Science 72

5. delete prn

> student$prn<-NULL

> student

name course marks

1 Pratikshya Maths 95

2 Pratik English 85

3 Vivan Science 72

4 Pari social 98

6. change records value marks=45

> student[1,3]=45

> student

name course marks

1 Pratikshya Maths 45

2 Pratik English 85

3 Vivan Science 72

4 Pari social 98

#create origin dataframe(

> producers<-data.frame(

+ surname=c("Spielberg","Scorsese","Hitchcock","Tarantino","Polansh"),

+ nationality=c("US","US","UK","US","POLAND"),

+ stringsAsFactors=FALSE)

> producers

surname nationality

1 Spielberg US

2 Scorsese US

3 Hitchcock UK

4 Tarantino US

5 Polansh POLAND

> movies<-data.frame(

+ surname=c("Spielberg","Scorsese","Hitchcock","Hitchcock","Spielberg","Tarantino","Polansh"),

+ title=c("Super8","Taxi driver","Psycho","North ny Northwest","Catch Me If You can","Reservoir Dogs","Chinatown"),

+ stringsAsFactors=FALSE)

> movies

surname title

1 Spielberg Super8

2 Scorsese Taxi driver

3 Hitchcock Psycho

4 Hitchcock North ny Northwest

5 Spielberg Catch Me If You can

6 Tarantino Reservoir Dogs

7 Polansh Chinatown

\*\*\*\*MERGING

> m1<-merge(producers,movies,by.x="surname")

> m1

surname nationality title

1 Hitchcock UK Psycho

2 Hitchcock UK North ny Northwest

3 Polansh POLAND Chinatown

4 Scorsese US Taxi driver

5 Spielberg US Super8

6 Spielberg US Catch Me If You can

7 Tarantino US Reservoir Dogs

CREATE DATA FRAME EMP AS

emp\_id and designation

another data frame HR with emp\_id and Salary

Merge two data frames using "emp\_id"

> Emp<-data.frame(

+ emp\_id=c(1,2,3,4,5),

+ designation=c("Clerk","Accountant","AdminOfficer","Non-teaching staff","Teaching staff"),

+ stringsAsFactors = FALSE)

> Emp

emp\_id designation

1 1 Clerk

2 2 Accountant

3 3 AdminOfficer

4 4 Non-teaching staff

5 5 Teaching staff

> HR<-data.frame(

+ emp\_id=c(1,2,3,4,5),

+ salary=c(5000,15000,25000,12500,35000),

+ stringsAsFactors = FALSE)

> HR

emp\_id salary

1 1 5000

2 2 15000

3 3 25000

4 4 12500

5 5 35000

> m2<-merge(Emp,HR,by.x="emp\_id")

> m2

emp\_id designation salary

1 1 Clerk 5000

2 2 Accountant 15000

3 3 AdminOfficer 25000

4 4 Non-teaching staff 12500

5 5 Teaching staff 35000

> print(seq(20,44)) =># print numbers from 20 to 44

[1] 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44

> print(mean(20:80)) => #print the mean

[1] 50

> print(sum(40:60)) => #print the sum

[1] 1050

#List Creation

#Vector with numeric from 1 upto 5

vect<-1:5

#Create a matrix

> mat<-matrix(1:10,ncol=5)

> dim(mat)

[1] 2 5

> mat

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

[1,] 1 3 5 7 9

[2,] 2 4 6 8 10

#create dataframe

> df<-data.frame(f1=c(23,45,2,3))

> df

f1

1 23

2 45

3 2

4 3

mylist1<-list(vect,mat,df)

mylist1

[[3]]

f1

1 23

2 45

3 2

4 3

#displaying element from list

> mylist1[[2]]

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

[1,] 1 3 5 7 9

[2,] 2 4 6 8 10

> mylist1[[1]]

[1] 1 2 3 4 5

> mylist1[[3]]

f1

1 23

2 45

3 2

4 3

#length of the list

> length(mylist1)

[1] 3

> exam\_data<-data.frame(

+ name=c("Anastasia","Dima","Katherine","James","Emily","Pratikshya","Pratik","Vivan","Kevin","Jonas"),

+ score=c(12.5,9,4,8,4.5,6,2.5,7,9.9,12),

+ attempts=c(1,2,3,1,2,3,3,2,1,3),

+ qualify=c("yes","no","yes","no","yes","no","yes","no","yes","no")

+ )

> print(exam\_data)

name score attempts qualify

1 Anastasia 12.5 1 yes

2 Dima 9.0 2 no

3 Katherine 4.0 3 yes

4 James 8.0 1 no

5 Emily 4.5 2 yes

6 Pratikshya 6.0 3 no

7 Pratik 2.5 3 yes

8 Vivan 7.0 2 no

9 Kevin 9.9 1 yes

10 Jonas 12.0 3 no

> new\_list=split(exam\_data,seq(nrow(exam\_data)))

> print("Dataframe rows to a list:")

[1] "Dataframe rows to a list:"

> print(new\_list)

$`1`

name score attempts qualify

1 Anastasia 12.5 1 yes

$`2`

name score attempts qualify

2 Dima 9 2 no

$`3`

name score attempts qualify

3 Katherine 4 3 yes

$`4`

name score attempts qualify

4 James 8 1 no

$`5`

name score attempts qualify

5 Emily 4.5 2 yes

$`6`

name score attempts qualify

6 Pratikshya 6 3 no

$`7`

name score attempts qualify

7 Pratik 2.5 3 yes

$`8`

name score attempts qualify

8 Vivan 7 2 no

$`9`

name score attempts qualify

9 Kevin 9.9 1 yes

$`10`

name score attempts qualify

10 Jonas 12 3 no

#Genertaing numbers in list

> -5:5

[1] -5 -4 -3 -2 -1 0 1 2 3 4 5

**#CONDITIONAL STATEMNET**

x<-30

if(x>20)

{

print("x is integer")

}

[1] "x is integer"

**# ifelse**

> y=ifelse(x>15,1,0)=> 1 means yes condition satisfied . 0 means condition not satisfied.

> y

[1] 1 ==> x is greater than 15 so o/p is 1. if it was not greater than the o/p would be 0.

> y=ifelse(x>15,"y","n")

> y

[1] "y"

**# for loop**

x<-c(5,23,19,28,11,16)

count<-0

for(val in x)

{

if(val %% 2==0) count=count+1

}

print(count)

[1] 2

> if(val %% 2 == 0) count=count+1

> print(val)

[1] 16

> print(count)

[1] 2

###########

> z=ifelse(x %in% c(5,16),x\*2,ifelse(x%in% c(20,30),x\*3,x\*4))

> z

[1] 10 92 76 112 44 32

**########SWITCH CASE**

> vtr<-c(150,200,250,300,350,400)

> option<-"mean"

> switch(option,

+ "mean"=print(mean(vtr)),

+ "mode"=print(mode((vtr))),

+ "median"=print(median((vtr)))

+ )

[1] 275

**\*\*\*\*\*GETTING THE CURRENT DIRECTORY\*\*\*\*\*\*\*\***

> getwd()

[1] "C:/Users/Pratikshya Subedi/Documents"

**\*\*\*\*\*\*\*CHANGING THE DIRECTORY\*\*\*\*\*\*\*\*\***

setwd("d:/MyDoc")

**\*\*\*\*\*Build in data set available in R:\*\*\*\*\*\***

data()

**\*\*\*\*For max value of any data set \*\*\*\***

> up<-max(CO2$uptake) ==>CO2 is name of data set

> up

[1] 45.5

> retval<-subset(CO2,uptake==max(uptake))

> retval

Plant Type Treatment conc uptake

21 Qn3 Quebec nonchilled 1000 45.5

> retval1<-subset(CO2,Treatment=="chilled")

> retval1

> retval3<-subset(CO2,Treatment=="chilled" & Plant=="Qc1")

> retval3

> setwd("C:/ctit")

> getwd()

[1] "C:/ctit"

> data<-read.csv("info.csv")

> print(data)

id name salary start\_date dept

1 1 Rick 623.30 2012-01-01 IT

2 2 Dan 515.20 2013-09-23 Operations

3 3 Michelle 611.00 2014-11-15 IT

4 4 Ryan 729.00 2014-05-11 HR

5 5 Gary 843.25 2015-03-27 Finance

6 6 Nina 578.00 2013-05-21 IT

7 7 Simon 632.80 2013-07-30 Operations

8 8 Guru 722.50 2014-06-17 Finance

> head(data,6) => gives first 6 records

id name salary start\_date dept

1 1 Rick 623.30 2012-01-01 IT

2 2 Dan 515.20 2013-09-23 Operations

3 3 Michelle 611.00 2014-11-15 IT

4 4 Ryan 729.00 2014-05-11 HR

5 5 Gary 843.25 2015-03-27 Finance

6 6 Nina 578.00 2013-05-21 IT

> print(ncol(data))

[1] 5

> print(nrow(data))

[1] 8

> print(is.data.frame(data))

[1] TRUE

> head(data,3)

id name salary start\_date dept

1 1 Rick 623.3 2012-01-01 IT

2 2 Dan 515.2 2013-09-23 Operations

3 3 Michelle 611.0 2014-11-15 IT

> tail(data,3) => displays last 3 records

id name salary start\_date dept

6 6 Nina 578.0 2013-05-21 IT

7 7 Simon 632.8 2013-07-30 Operations

8 8 Guru 722.5 2014-06-17 Finance

> summary(data)

id name salary start\_date dept

Min. :1.00 Dan :1 Min. :515.2 2012-01-01:1 Finance :2

1st Qu.:2.75 Gary :1 1st Qu.:602.8 2013-05-21:1 HR :1

Median :4.50 Guru :1 Median :628.0 2013-07-30:1 IT :3

Mean :4.50 Michelle:1 Mean :656.9 2013-09-23:1 Operations:2

3rd Qu.:6.25 Nina :1 3rd Qu.:724.1 2014-05-11:1

Max. :8.00 Rick :1 Max. :843.2 2014-06-17:1

(Other) :2 (Other) :2

>View(data)

> x<-subset(data,as.Date(start\_date)>as.Date("2018-01-01"))

> x

[1] id name salary start\_date dept

<0 rows> (or 0-length row.names)

> x<-subset(data,as.Date(start\_date)>as.Date("2014-06-17"))

> x

id name salary start\_date dept

3 3 Michelle 611.00 2014-11-15 IT

5 5 Gary 843.25 2015-03-27 Finance

> y<-subset(data,salary==max(salary))

> y

id name salary start\_date dept

5 5 Gary 843.25 2015-03-27 Finance

> z<-subset(data,salary==min(salary))

> z

id name salary start\_date dept

2 2 Dan 515.2 2013-09-23 Operations

> list.files()

[1] "info.csv"

# 12th Dec,2019

> my.name<-readline(prompt="Enter name:")

Enter name:Pratikshya

> my.age<-readline(prompt="Enter age:")

Enter age:19

#convert character into integer

> my.age<-as.integer(my.age) # argument to be of integer type

> is.integer(my.age) #is my.age an interger? (check)

[1] TRUE

> print(paste("Hi,",my.name,"next year you will be",my.age+1,"years old."))

[1] "Hi, Pratikshya next year you will be 20 years old."

\*\*\*\*\*\*\*\*\* In R we use dot(.) while declareing variable name\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

########### DEFINING USER DEFINED FUNCTION

# Create a function to print cube of numbers in sequence.

> new.function<-function(a)

+ {

+ for(i in 1:a)

+ {

+ b<-i^3

+ print(b)

+ }

+ }

###### call function with argument

> new.function(5)

[1] 1

[1] 8

[1] 27

[1] 64

[1] 125

### Even and odd number

> even.function<-function(a)

+ {

+ ifelse(a %% 2==0,"even","odd")

+ }

> even.function(2)

[1] "even"

> even.function(5)

[1] "odd"

############

### Create a function without an argument.

> new.function<-function()

+ {

+ for(i in 1:5)

+ {

+ print(i^2)

+ }

+ }

# Call the function without supplying an argument.

> new.function()

[1] 1

[1] 4

[1] 9

[1] 16

[1] 25

##### Even and odd by taking user input

a.num<-readline(prompt="Enter number:")

a.num<-as.integer(a.num)

even.function<-function(a)

{

ifelse(a %% 2==0,"even","odd")

}

even.function(a.num)

OR

num<-readline(prompt="Enter number:")

num<-as.integer(num)

even.function<-function(num)

{

ifelse(num %% 2==0,"even","odd")

}

even.function()

## Create a function with arguments.

> new.function<-function(a=4,b=6)

+ {

+ result<-a\*b

+ print(result)

+ }

## call the function without giving any arguments.

> new.function()

[1] 24

# Call the function with giving new values of the arguments.

> new.function(9,5)

num1<-readline(prompt="Enter number:")

num1<-as.integer(num1)

num2<-readline(prompt="Enter number:")

num2<-as.integer(num2)

check.function<-function(num1,num2)

{

add<-num1+num2

print(paste("Addition",add))

sub<-num1-num2

print(paste("Subtraction",sub))

pro<-num1\*num2

print(paste("Multiplication",pro))

div<-num1/num2

print(paste("Division",div))

}

check.function(num1,num2)

[1] "Addition 15"

[1] "Subtraction 5"

[1] "Multiplication 50"

[1] "Division 2"

[1] 45

# 19th dec,2019

In xlsx

#to read a file

info<-read.xlsx("filename",sheetIndex=1)

######

> x<-seq(-pi,pi,0.1)

> x

[1] -3.14159265 -3.04159265 -2.94159265 -2.84159265 -2.74159265 -2.64159265

[7] -2.54159265 -2.44159265 -2.34159265 -2.24159265 -2.14159265 -2.04159265

[13] -1.94159265 -1.84159265 -1.74159265 -1.64159265 -1.54159265 -1.44159265

[19] -1.34159265 -1.24159265 -1.14159265 -1.04159265 -0.94159265 -0.84159265

[25] -0.74159265 -0.64159265 -0.54159265 -0.44159265 -0.34159265 -0.24159265

[31] -0.14159265 -0.04159265 0.05840735 0.15840735 0.25840735 0.35840735

[37] 0.45840735 0.55840735 0.65840735 0.75840735 0.85840735 0.95840735

[43] 1.05840735 1.15840735 1.25840735 1.35840735 1.45840735 1.55840735

[49] 1.65840735 1.75840735 1.85840735 1.95840735 2.05840735 2.15840735

[55] 2.25840735 2.35840735 2.45840735 2.55840735 2.65840735 2.75840735

[61] 2.85840735 2.95840735 3.05840735

> plot(x,sin(x)) => to get the graph

### Adding titles and labeling axes

plot(x,sin(x), main="The Sine Function", ylab="sin(x)")

> plot(x,sin(x), main="The Sine Function", ylab="sin(x)",type="l",col="blue")

#change the plot type with the arguments type.

1. "p" - points

2. "l"- lines

3. "b"- both points and lines

4."c" - empty points joined by lines

5."o" - overplotted points and lines

6. "s" and "S" - stair steps

7. "h" - histogram-like vertical lines

8. "n" - does not produce any points or lines

Eg;

plot(x,sin(x))

> plot(x,sin(x), main="The Sine Function", ylab="sin(x)")

> plot(x,sin(x), main="The Sine Function", ylab="sin")

> plot(x,sin(x), main="The Sine Function", ylab="sin(x)",type="l",col="blue")

> plot(x,sin(x), main="The Sine Function", ylab="sin(x)",type="n",col="red")

> plot(x,sin(x), main="The Sine Function", ylab="sin(x)",type="b",col="red")

> plot(x,sin(x), main="The Sine Function", ylab="sin(x)",type="c",col="green")

> plot(x,sin(x), main="The Sine Function", ylab="sin(x)",type="o",col="green")

> plot(x,sin(x), main="The Sine Function", ylab="sin(x)",type="s",col="yellow")

> plot(x,sin(x), main="The Sine Function", ylab="sin(x)",type="h",col="yellow")

Ex:2

v<-c(7,12,28,3,41)

> plot(v,type="o",col="red")

## Plot the bar chart

> plot(v,type="o",col="red",xlab="Month",ylab="Rainfall",main="Rain fall chart")

####multiple data line

> v<-c(7,12,28,3,41)

> t<-c(14,7,6,19,3)

> plot(v,type="o",col="red",xlab="Month",ylab="Rain fall",main="Rain fall chart")

> lines(t,type="o",col="blue")

#2nd Jan,2020

plot(cars, type="s",col="blue",xlab = "Speed (mph)", ylab = "Stopping distance (ft)",las = 1, xlim = c(0, 25))

\*\*cars

plot(cars$speed,type="s",col="orange",xlab="speed",ylab="dist")

####Create scatterplot

input<-mtcars[,c('wt','mpg')]

print(head(input))

plot(type="s",col="blue",x=input$wt,y=input$mpg,xlab="Weight",ylab="Milage",xlim=c(2.5,5),ylim=c(15,30),main="Weight vs Milage")

#### Create data for the graph

x<-c(21,62,10,53)

lables<-c("London","New York","Sinagpore","Mumbai")

##### Give the chart file a name

png(file="a.jpg") ==>to save the plots

##

x<-c(21,62,10,53)

> labels<-c("London","New York","Sinagpore","Mumbai")

> pie(x,labels)

> pie(x,labels,main="City pie chart",col=rainbow(length(x)))

> x1=pie(x,labels,main="City pie chart",col=rainbow(length(x)))

> x1

# 6th jan 2020

#cars from datasets

###Create a piechart for cars

pie(cars)

pie(cars,main="Cars",col=rainbow(length(cars)),labels=c("Mon","Tue","Wed","Thu","Fri"))

#Define some colours ideal for black & white print

colors<-c("Mon","Tue","Wed","Thu","Fri")

#Concatenate a "%" char after each value.

cars\_labels<-paste(car\_lebels,"%",sep="")

# DOTCHART

dotchart(cars)

dotchart((cars),color=c("red","blue","green"),main="Dotchart for Autos",cex=0.8) ==>cex is used for changing the value of dots.

#Make empty chart

plot(1,1,xlim=c(1,5.5),ylim=c(0,7),type="n",ann=FALSE)

#Plot digits 0-4 with increasing size and color

text(1:5,rep(6,5),label=c(0:4),cex=1:5,col=1:5) ==> text(1:5 numeric vectors of co-ordinates where the text labels should be written rep is starting location of value is written.

#Plot symbols 0-4 with increasing size and color

points(1:5,rep(5,5),cex=1:5,col=1:5,pch=0.4) ==> pch plotting "character", i.e symbols to use.

# 9th Jan

plot(1,1,xlim=c(1,5.5),ylim=c(0,7),type="n",ann=FALSE)

text(1:5,rep(6,5),label=c(0:4),cex=1:5,col=1:5) ==> rep is for X-axis and Y-axis value

points(1:5,rep(5,5),cex=1:5,col=1:5,pch=0.4)

points(1:5,rep(5,5),cex=1:5,col=1:5,pch=8) ==> change pch for which symbol you want

text((1:5)+0.4,rep(5,5),cex=0.6,(0:4)) ==> cex is character or symbol expansion

#Plot Symbol 5-9 with labels

points(1:5,rep(4,5),cex=2,pch=(5:9))

text((1:5)+0.4,rep(3,5),cex=0.6,(10:14))

# Plot Symbol 10-14 with labels

points(1:5,rep(4,5),cex=2,pch=(10:14))

text((1:5)+0.4,rep(3,5),cex=0.6,(15:19))

# #############

# Bar plot

#Simple Bar Plot

counts<-table(mtcars$gear)

barplot(counts,main="Car Distribution",xlab="Number of Gears")

######## Stacked Bar Plot

counts<-table(mtcars$gear)

barplot(counts,main="Car Distribution",xlab="Number of Gears",col=c("red","blue"),legend=rownames(counts))

######### Use Car dataset

###Find out correlation between two variable speed and distance

cor(cars$speed,cars$dist) ###calculate correlation between speed and distance

> cor(cars$speed,cars$dist)

[1] 0.8068949

#Building the linear Regression Model

## Linear regression is used to predict the value of a value of a continuous variable Y based on one or more input predictor variables X.

#### That is Distance(dist) as a function for speed.

> linearMod<-lm(dist~speed,data=cars)

> print(linearMod)

Call:

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

Coefficients:

(Intercept) speed

-17.579 3.932

# 13th Jan,2020

#Scatter plot

scatter.smooth(x=cars$speed,y=cars$dist,main="Dist~Speed")

### Using BoxPlot To Check For Outliers

par(mfrow=c(1,2))

boxplot(cars$speed,main="Speed",sub=paste("Outlier rows:",boxplot.stats(cars$speed)$out))

boxplot(cars$dist,main="Distance",sub=paste("Outlier rows:",boxplot.stats(cars$dist)$out))

CO2

boxplot(CO2$conc,main="CONC",sub=paste("Outlier rows:",boxplot.stats(CO2$conc)$out))

boxplot(CO2$uptake,main="UPTAKE",sub=paste("Outlier rows:",boxplot.stats(CO2$uptake)$out))

######FUNCTIONS IN R

> x<-"learning R"

> s<-"hello"

> y<-toupper(x) ==> Converts to upper case

> y

[1] "LEARNING R"

> x1<-tolower(s) ==> converts to lower case

> x1

[1] "hello"

> x2<-substr(s,0,4)

> x2

[1] "hell"

> x3<-strsplit(x,"r")

> x3

[[1]]

[1] "lea" "ning R"

> x4<-paste(x,"String Functions",sep="")

> x4

[1] "learning RString Functions"

> x5<-sub("learning","examples",x)

> x5

[1] "examples R"

> res<-format(13.123456789,digits=7)

> print(res)

[1] "13.12346"

> res<-format(c(6,25.14521),scientific=TRUE)

> print(res)

[1] "6.000000e+00" "2.514521e+01"

> res<-format(13.57,nsmall=5)

> print(res)

[1] "13.57000"

> res1<-format(16)

> print(res1)

[1] "16"

re1<-format(49.7,width=6) ==> numbers are padded with blank sspace in beginning for width

print(re1)

[1] " 49.7"

#16th Jan,2020

install.packages("tm") # for text mining

library("tm")

text<-readlines(file.choose()) # load the files

# Read the text files from internet

### file path<-"http://filename.txt"

#text<-readlines(filePath)

=> text

docs<-corpus(VectorSource(text))

inspect(docs)

text<-c(docs)

#docs\_ids<-c(1)

df<-dat.frame(docs\_id=doc\_ids,text=text,stringsAsfactors=FALSE)

df\_corpus<-Corpus(DataframeSource(df))

## Text transformation

###toSpace<-content\_transformer(function(x,pattern)gsub(pattern," "))

#docs<-tm\_map(docs,toSpace,"/")

##docs<-tm\_map(docs,toSpace,"@")

###docs<-tm\_map(docs,toSpace,"\\|")

docs

# Convert the text to lowercase

docs<-tm\_map(docs,content\_transformer(tolower))

# Remove numbers

docs<-tm\_map(docs,removeNumbers)

#Remove english common Stopwords

docs<-tm\_map(docs,removeWords,stopwords("english"))

#Remove your own stopword

#Specify your stopwords as a character vector

docs<-tm\_map(docs, removeWords,c("the","abc"))

#Remove punctuations

docs<-tm\_map(docs,stripWhitespace)

# Text stemming

docs<-tm\_map(docs,stemDocument)

#Remove stop Words

docs<-tm\_map(docs,removeWords,stopwords("english"))

### Build a term document matrix

dtm<-TermDocumentmatrix(docs)

m<-as.matrix(dtm)

v<-sort(rowSums(m),decreasing=TRUE)

d<-data.frame(word=names(v),freq=v)

head(d,10)

# 23rd Jan,2020

##### Generate the Word Cloud

set.seed(1234) ### Use the set.seed function when running simulations to ensure all results, figures, etc are reproducible

wordcloud(words=d$word,freq=d$freq,min.freq=1,max.words=2000, random.order=FALSE, rot.per=0.35, color=brewer.pal(8,"dark2")

## Explore frequent terms and their associations

findFreqTerms(dtm,lowfreq=4)

### analyze the association between frequent terms

##(i.e,terms which correlate) using findAssocs() function

findAssocs(dtm,terms="freedom", corlimit=0.3)

## frequency table of words

head(d,10)

## Plot word frequencies

barplot(d[1:10,]$freq, las=2, names.arg=d[1:10,] $word, col="lightblue", main="Most frequent words", ylab="Word frequencies")