MITHUN G
19BTRCR006

R PROGRAMMING LANGUAGE LAB

Name:- Mithun G

USN:- 19BTRCR006

Branch: - Data Science

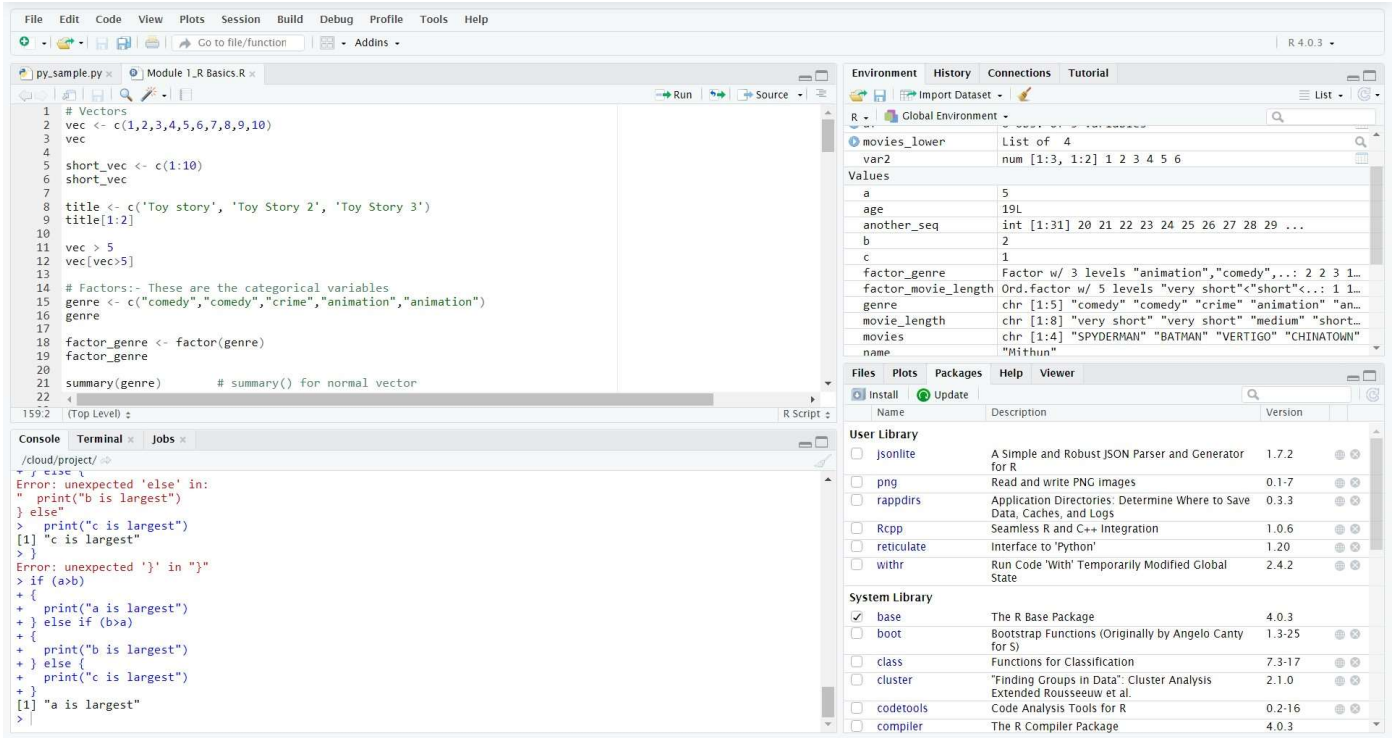
Subject Code:- 18CSI401L

Name:-Mithun G

USN:-19BTRCR006

LAB PROGRAM 1

1. Install and configure R, set working directory.
2. Install Packages and calling installed packages
3. R studio environment and functionalities of R studio
4. Explore assignment, comments, and case sensitivity in R.
5. Use some built-in functions such as pi, sqrt, log, round, factorial, trigonometric functions, ect. From R.
6. Create R script files and call it from R console.
7. Explore for getting help, examples, and demonstrations on R.
8. Use R as calculator.
9. Explore different mathematical operators.



Name:- Mithun G

USN:- 19BTRCR006

LAB PROGRAM 2

1. Explore assignment operator

In [2]:

```
a <- 5
a
```

5

In [3]:

```
b = 5
b
```

5

In [6]:

```
3 -> c
c
```

3

In [7]:

```
d <<- 4
d
```

4

In [9]:

```
4 ->> e
e
```

4

2.Create vectors using c(), seq(), rep(), colon operator.

In [17]:

```
vec <- c(1,2,3)
vec
```

1 · 2 · 3

In [19]:

```
vec1 <- seq(1,10,2)
vec1
```

1 · 3 · 5 · 7 · 9

In [21]:

```
vec2 <- rep(1,5)
vec2
```

1 · 1 · 1 · 1 · 1

In [27]:

```
vec3 <- c(1:10)
vec3
```

1 · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9 · 10

3. Create different matrices using matrix() operator and explore its rows, columns, and diagonals.

In [47]:

```
mat_r <- matrix(data=c(1:9),nrow=3)
mat_r
```

A matrix:
3 × 3 of
type int

1	4	7
2	5	8
3	6	9

In [48]:

```
mat_r[1,1:3]      # printing the first row elements
```

1 · 4 · 7

In [53]:

```
mat_c <- matrix(data=c(1:9),nrow=3, byrow=T)
mat_c
```

A matrix:
3 × 3 of
type int

1	2	3
4	5	6
7	8	9

In [50]:

```
mat_c[1:3,1]      # printing the first column elements
```

1· 4· 7

In [54]:

```
diag(mat_c)     # printing the diagonal elements
```

1· 5· 9

4. Perform different basic operation of matrices on above created matrices.

In [59]:

```
# previously created matrices
print(mat_r)
print(mat_c)
```

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

In [58]:

```
# matirx addition
mat_r + mat_c
```

A matrix: 3 × 3
of type int

2	6	10
6	10	14
10	14	18

In [60]:

```
# matirx subtraction
mat_r - mat_c
```

A matrix: 3
× 3 of type
int

0	2	4
-2	0	2
-4	-2	0

In [61]:

```
# matirx hadamard multiplication
mat_r * mat_c
```

A matrix: 3 × 3
of type int

1	8	21
8	25	48
21	48	81

In [62]:

```
# matirx true multiplication
mat_r %*% mat_c
```

A matrix: 3 × 3 of
type dbl

66	78	90
78	93	108
90	108	126

In [64]:

```
# scalar multiplication
(1/5) * mat_r
```

A matrix: 3 × 3

of type dbl

0.2	0.8	1.4
0.4	1.0	1.6
0.6	1.2	1.8

5. Create single and multidimensional arrays with array() command.

In [65]:

```
# single dimesnional array
arr <- array(1:10)
arr
```

1· 2· 3· 4· 5· 6· 7· 8· 9· 10

In [70]:

```
# multi dimesnional array
arr_md= array(data=1:3, dim= c(2,4))
arr_md
```

A matrix: 2 ×

4 of type int

1	3	2	1
2	1	3	2

6. Explore length(), dim(), ncol(), nrow() operators on above matrices and arrays.

In [82]:

```
print(mat_r)   # previously created matrix
print(arr_md)  # previously created array
```

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

In [83]:

```
# length
print(length(mat_r))
print(length(arr_md))
```

```
[1] 9
[1] 8
```

In [84]:

```
# dimensions
print(dim(mat_r))
print(dim(arr_md))
```

```
[1] 3 3
[1] 2 4
```

In [85]:

```
# ncol() and nrow()
print(ncol(mat_r))
print(nrow(arr_md))
```

```
[1] 3
[1] 2
```

7. Explore commands for Selecting and extracting elements from above matrices and arrays.

In [86]:

```
print(mat_r)   # previously created matrix
print(arr_md)  # previously created array
```

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

In [87]:

```
# selecting element 5 from matrix
mat_r[2,2]
```

```
5
```

In [89]:

```
# selecting element 3 from 1st row from array
arr_md[1,2]
```

```
3
```

In [91]:

```
# extracting 1st row of the matrix
mat_r[1,1:3]
```

1 · 4 · 7

In [98]:

```
# extracting 1st row of the array
arr_md[1,]
```

1 · 3 · 2 · 1

8. Explore logical operators from R programming language.

In [117]:

```
vec1 <- c(T,F,T,F)
vec2 <- c(F,F,T,T)
print(vec1)
print(vec2)
```

```
[1]  TRUE FALSE  TRUE FALSE
[1] FALSE FALSE  TRUE  TRUE
```

In [118]:

```
# & operation
vec1 & vec2
```

FALSE · FALSE · TRUE · FALSE

In [119]:

```
# | operation
vec1 | vec2
```

TRUE · FALSE · TRUE · TRUE

In [115]:

```
# && operation
vec1 && vec2
```

FALSE

In [120]:

```
# || operation
vec1 || vec2
```

TRUE

In [121]:

```
# not operation
!vec1
```

FALSE · TRUE · FALSE · TRUE

9. Remove elements from selected positions from a considered matrix.

In [132]:

```
print(mat_c)  # old matrix created
```

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

In [135]:

```
# removing element 2
mat_c[-4]   # -4 because 2 is at 4th position on matrix
```

1 · 4 · 7 · 5 · 8 · 3 · 6 · 9

In [141]:

```
# removing first 2 elements
mat_c[-1:-2]
```

7 · 2 · 5 · 8 · 3 · 6 · 9

Name:- Mithun

USN:- 19BTRCR006

LAB PROGRAM 3

1. Create some complex data structure variables such as list and data frames using list() and data.frame commands.

In [1]:

```
# creating a dataframe
df <- data.frame(int_col= c(1:5), double_col=c(5,5.1,5.2,5.3,5.4))
df
```

A data.frame: 5 × 2

int_col <int>	double_col <dbl>
1	5.0
2	5.1
3	5.2
4	5.3
5	5.4

In [2]:

```
# creating a list
list_var <- list(1:10,11:20)
print(list_var)
```

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

[[2]]
 [1] 11 12 13 14 15 16 17 18 19 20
```

2. Create data using data.frames, lists, and tables.

In [3]:

```
# Creating a dataframe using data.frame
let <- letters[1:5]     # generating english letters, for capital letters use LETTERS
d <- data.frame(x = 1, y = 1:10, letters=let)     # factors means categorical variables
d
```

A data.frame: 10 × 3

x	y	letters
<dbl>	<int>	<fct>
1	1	a
1	2	b
1	3	c
1	4	d
1	5	e
1	6	a
1	7	b
1	8	c
1	9	d
1	10	e

In [4]:

```
# Creating a list
let <- letters[1:5]
l <- list(1:5, let)
print(l)
```

```
[[1]]
[1] 1 2 3 4 5

[[2]]
[1] "a" "b" "c" "d" "e"
```

In [5]:

```
# importing iris dataset
iris <- datasets::iris
head(iris)
```

A data.frame: 6 × 5

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
	<dbl>	<dbl>	<dbl>	<dbl>	<fct>
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

In [6]:

```
# Creating a table and counting the values from the iris dataset
t <- table(iris$Species, iris$Petal.Width)
t
```

```
            0.1 0.2 0.3 0.4 0.5 0.6  1 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9
2
  setosa      5  29   7   7   1   1  0   0   0   0   0   0   0   0   0   0
0
  versicolor  0   0   0   0   0   0  7   3   5  13   7  10   3   1   1   0
0
  virginica   0   0   0   0   0   0  0   0   0   0   1   2   1   1  11   5
6

            2.1 2.2 2.3 2.4 2.5
  setosa      0   0   0   0   0
  versicolor  0   0   0   0   0
  virginica   6   3   8   3   3
```

3. Implement basic R operations (data input, missing values, Importing data into R using different formats : xlsx, CSV, Text files).

- use read.text for reading a dataset of format .txt
- use read.excel for reading a dataset of format .xlsx

In [9]:

```
# importing a csv file to a dataframe
data <- read.csv('titanictrain.csv')
head(data)
```

A data.frame: 6 × 12

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
	<int>	<int>	<int>	<fct>	<fct>	<dbl>	<int>	<int>	<fct>	<dbl>
1	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2500
2	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	71.2833
3	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.9250
4	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1000
5	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.0500
6	6	0	3	Moran, Mr. James	male	NA	0	0	330877	8.4583

In [10]:

```
sum(is.na(data))
```

177

In [11]:

```
sum(is.na(data$Age))   # age feature is having all the 177 missing values
```

177

In [12]:

```
# removing the NA values
data_new <- na.omit(data)
sum(is.na(data_new))
```

0

In [13]:

```
cat("Rows before removing the NA values:", nrow(data), "\n")  # shape before removing the N
cat("Rows after removing the NA values: ", nrow(data_new) )  #   (891-177)
```

```
Rows before removing the NA values: 891
Rows after removing the NA values:  714
```

In [14]:

```
# filling the missing values with mean
data[is.na(data)]= mean(data_new$Age)
sum(is.na(data))
```

0

4. Explore data type conversions from one data structure to another with commands such as as.data.frame(), as.vector(), is.data.frame(), is.vector; and find the data type with class() command.

In [15]:

```
head(data)    # a dataframe
```

A data.frame: 6 × 12

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fa
	<int>	<int>	<int>	<fct>	<fct>	<dbl>	<int>	<int>	<fct>	<db
1	1	0	3	Braund, Mr. Owen Harris	male	22.00000	1	0	A/5 21171	7.25(
2	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38.00000	1	0	PC 17599	71.28:
3	3	1	3	Heikkinen, Miss. Laina	female	26.00000	0	0	STON/O2. 3101282	7.92:
4	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.00000	1	0	113803	53.10(
5	5	0	3	Allen, Mr. William Henry	male	35.00000	0	0	373450	8.05(
6	6	0	3	Moran, Mr. James	male	29.69912	0	0	330877	8.45:

In [16]:

```
# is.data.frame()
is.data.frame(data)
```

TRUE

In [24]:

```
# converting a list into vector
list_vec <- list(1:10)
data_vec= as.vector(list_vec)
is.vector(data_vec)
```

TRUE

In [27]:

```
# converting a vector into dataframe
data_new <- as.data.frame(data_vec)
is.data.frame(data_new)
```

TRUE

5. Explore function programming in R.

In [37]:

```
# Printing the squares
square_func <- function(a)
{
  for(i in 1:a)
  {
    b <- i^2
    print(b)
  }
}
square_func(6)
```

```
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
[1] 36
```

6. Explore loops in R programming such as if-else-ifelse, for, while, repeat-break, etc

In [38]:

```
# for loop
cubic_func <- function(a)
{
  for(i in 1:a)
  {
    b <- i*3
    print(b)
  }
}
cubic_func(6)
```

```
[1] 3
[1] 6
[1] 9
[1] 12
[1] 15
[1] 18
```

In [39]:

```
# while loop
power<-2
i<-1
while(i<=5){
  print(power**i)
  i=i+1
}
```

```
[1] 2
[1] 4
[1] 8
[1] 16
[1] 32
```

In [48]:

```
# if-else conditional statements
num1 <- 3
num2 <- 6
num3 <- 9
if(num1>num2 && num1>num3){
    max = num1
} else if(num2>num1 && num2>num3){
    max = num2
} else{
    max =num3
}

print(max)
```

```
[1] 9
```

In [51]:

```
# repeat-break
x <- 1
repeat {
    print(x)
    x = x+1
    if (x == 6){
        break
}
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

Name:- Mithun G

USN:-19BTRCR006

LAB PROGRAM 4

1. Calculate the interest earned after 5 years on an investment of $2000, assuming an interest rate of 3% compounded annually.

In [20]:

```
# Interest earned
CI = 2000*(1+(3/100))^5
cat("Total amount:", CI)
cat("\nInterest earned:",CI - 2000)
```

```
Total amount: 2318.548
Interest earned: 318.5481
```

2. Use R to calculate the area of a circle with radius 7 cm.

In [22]:

```
area = pi*(7/100)*(7/100)
cat("Area of circle in m:", area)
```

```
Area of circle in m: 0.0153938
```

3. Do you think there is a difference between 48:14^2 and 48:(14^2)?

In [27]:

```
a <- 48:14^2
a
```

```
 48 · 49 · 50 · 51 · 52 · 53 · 54 · 55 · 56 · 57 · 58 · 59 · 60 · 61 · 62 ·
 63 · 64 · 65 · 66 · 67 · 68 · 69 · 70 · 71 · 72 · 73 · 74 · 75 · 76 · 77 ·
 78 · 79 · 80 · 81 · 82 · 83 · 84 · 85 · 86 · 87 · 88 · 89 · 90 · 91 · 92 ·
 93 · 94 · 95 · 96 · 97 · 98 · 99 · 100 · 101 · 102 · 103 · 104 · 105 · 106 ·
 107 · 108 · 109 · 110 · 111 · 112 · 113 · 114 · 115 · 116 · 117 · 118 · 119 ·
 120 · 121 · 122 · 123 · 124 · 125 · 126 · 127 · 128 · 129 · 130 · 131 · 132 ·
 133 · 134 · 135 · 136 · 137 · 138 · 139 · 140 · 141 · 142 · 143 · 144 · 145 ·
 146 · 147 · 148 · 149 · 150 · 151 · 152 · 153 · 154 · 155 · 156 · 157 · 158 ·
 159 · 160 · 161 · 162 · 163 · 164 · 165 · 166 · 167 · 168 · 169 · 170 · 171 ·
 172 · 173 · 174 · 175 · 176 · 177 · 178 · 179 · 180 · 181 · 182 · 183 · 184 ·
 185 · 186 · 187 · 188 · 189 · 190 · 191 · 192 · 193 · 194 · 195 · 196
```

In [26]:

```
b <- 48:(14^2)
b
```

```
 48 ·  49 ·  50 ·  51 ·  52 ·  53 ·  54 ·  55 ·  56 ·  57 ·  58 ·  59 ·  60 ·  61 ·  62 ·
 63 ·  64 ·  65 ·  66 ·  67 ·  68 ·  69 ·  70 ·  71 ·  72 ·  73 ·  74 ·  75 ·  76 ·  77 ·
 78 ·  79 ·  80 ·  81 ·  82 ·  83 ·  84 ·  85 ·  86 ·  87 ·  88 ·  89 ·  90 ·  91 ·  92 ·
 93 ·  94 ·  95 ·  96 ·  97 ·  98 ·  99 ·  100 ·  101 ·  102 ·  103 ·  104 ·  105 ·  106 ·
 107 ·  108 ·  109 ·  110 ·  111 ·  112 ·  113 ·  114 ·  115 ·  116 ·  117 ·  118 ·  119 ·
 120 ·  121 ·  122 ·  123 ·  124 ·  125 ·  126 ·  127 ·  128 ·  129 ·  130 ·  131 ·  132 ·
 133 ·  134 ·  135 ·  136 ·  137 ·  138 ·  139 ·  140 ·  141 ·  142 ·  143 ·  144 ·  145 ·
 146 ·  147 ·  148 ·  149 ·  150 ·  151 ·  152 ·  153 ·  154 ·  155 ·  156 ·  157 ·  158 ·
 159 ·  160 ·  161 ·  162 ·  163 ·  164 ·  165 ·  166 ·  167 ·  168 ·  169 ·  170 ·  171 ·
 172 ·  173 ·  174 ·  175 ·  176 ·  177 ·  178 ·  179 ·  180 ·  181 ·  182 ·  183 ·  184 ·
 185 ·  186 ·  187 ·  188 ·  189 ·  190 ·  191 ·  192 ·  193 ·  194 ·  195 ·  196
```

Answer:-There is no difference between a and b, as both results in same output

4. Using rep() and seq()as needed, create the vectors:

000001111122222333334444 and 123451234512345123451234512345

In [44]:

```
rep(seq(0,4),each=5, len=25)
```

```
 0 ·  0 ·  0 ·  0 ·  0 ·  1 ·  1 ·  1 ·  1 ·  1 ·  2 ·  2 ·  2 ·  2 ·  2 ·  3 ·  3 ·  3 ·  3 ·
 3 ·  4 ·  4 ·  4 ·  4 ·  4
```

In [47]:

```
rep(seq(1,5),each=1, len=25)
```

```
 1 ·  2 ·  3 ·  4 ·  5 ·  1 ·  2 ·  3 ·  4 ·  5 ·  1 ·  2 ·  3 ·  4 ·  5 ·  1 ·  2 ·  3 ·  4 ·
 5 ·  1 ·  2 ·  3 ·  4 ·  5
```

5. Create the vector

[1]000111100011110001111000111100011111

and convert it to a factor. Identify the levels of the result, and then change the level labels to obtain the factor:
[1] Male Male Male Female Female Female Female Male Male

[10] Male Female Female Female Female Male Male Male Female

[19] Female Female Female Male Male Male Female Female Female

[28] Female Male Male Male Female Female Female Female

Levels: Male Female

In [56]:

```
vec <- rep(c(rep(0,each = 3),rep(1, each = 4)), 5)
vec
```

0 · 0 · 0 · 1 · 1 · 1 · 1 · 0 · 0 · 0 · 1 · 1 · 1 · 1 · 0 · 0 · 0 · 1 · 1 ·
1 · 1 · 0 · 0 · 0 · 1 · 1 · 1 · 1 · 0 · 0 · 0 · 1 · 1 · 1 · 1

In [58]:

```
vec1 <- factor(vec, levels = c(0,1), labels = c("Male","Female") )
vec1
```

Male · Male · Male · Female · Female · Female · Female · Male · Male · Male ·
Female · Female · Female · Female · Male · Male · Male · Female · Female ·
Female · Female · Male · Male · Male · Female · Female · Female · Female ·
Male · Male · Male · Female · Female · Female · Female

▼ **Levels**:

'Male' · 'Female'

6. Use more.colors vector, rep() and seq() to create the vector

"red" "yellow" "blue" "yellow" "blue" "green" "blue" "green" "magenta" "green" "magenta" "cyan"

In [62]:

```
colors <- c("red", "yellow", "blue","green","magenta","cyan")
colors[rep(seq(1:3),times=4) + rep(0:3, each=3)]
```

'red' · 'yellow' · 'blue' · 'yellow' · 'blue' · 'green' · 'blue' · 'green' · 'magenta' ·
'green' · 'magenta' · 'cyan'

Name: Mithun G

USN:19BTRCR006

LAB PROGRAM 5

1. Assume 4- binary digit accuracy for the following computations.
a) Write out the binary representation for the approximate value of 6/7.
b) Write out the binary representation for the approximate value of 1/7.
c) Add the two binary representations obtained above, and convert back to the decimal representation.
d) Compare the result of part (c) with the result from adding the binary representations of 6 and 1, followed by division by the binary representation of 7.

install.packages("R.utils")

In [63]:

```
library(R.utils)
#install.packages("stringr")
# library(stringr)
```

In [74]:

```
a <- 6/7
float2Bin<- function(x)
{
  int_part<- floor(x)
  dec_part<- x-int_part
  int2bin<- intToBin(int_part)
  dec2bin<- str_pad(intToBin(dec_part*2^31),31,pad='0') #O.o
  paste0(int2bin,".",dec2bin)
}
float2Bin(a)
```

'0.11011011011011011011011011011101'

In [75]:

```
b<- 1/7
float2Bin(b)
```

'0.00100100100100100100100100100010'

In [79]:

```
sum = a + b
answer <- as.binary(sum)
answer
```

1

In [127]:

```
new_a <- as.binary(6)
new_b <- as.binary(1)
new_sum <- new_a + new_b

# Just comparing the answer in (c) with the binary sum of 6 + 1
answer   # previous answer
new_sum  # binary sum of 6 and 1
```

```
1

1  1  1
```

In [134]:

```
# followed by division of binary number of 7
divided_new_sum = as.integer(new_sum) / 7
as.binary(divided_new_sum)
```

```
1
```

2. In R, evaluate the expressions
2^52 + k – 2^52
2^53 + k – 2^53
2^54 + k – 2^54,
for the cases where k = 1, 2, 3, 4. Explain what you observe. What could be done to obtain results in R which are mathematically correct?

Answer We can make use of functions

In [164]:

```
# before getting correctly
func <- function(k){
print(2^52 + k - 2^52)
print((2^53) + k - (2^53))
print(2^54 + k - 2^54)
}

func(c(1,2,3,4))
```

```
[1] 1 2 3 4
[1] 0 2 4 4
[1] 0 0 4 4
```

```
install.packages('bit64')
```

```
library(bit64)
```

In [39]:

```
# after manipualting to integer64
a <- as.integer64(2**52)
func <- function(k){
print(a + k - a)
print(a*2 + k - a*2)
print(a*4 + k - a*4)
}

func(c(1,2,3,4))
```

```
integer64
[1] 1 2 3 4
integer64
[1] 1 2 3 4
integer64
[1] 1 2 3 4
```

3. The following are a sample of observations on incoming solar radiation at a greenhouse:
11.1 10.6 6.3 8.8 10.7 11.2 8.9 12.2
a) Assign the data to an object called solar.radiation.
b) Find the mean, median, range, and variance of the radiation observations.
c) Add 10 to each observation of solar.radiation, and assign the result to sr10. Find the mean, median, range, and variance of sr10. Which statistics change, and by how much?
d) Plot a histogram of the solar.radiation, sr10.

In [90]:

```
solar.radiation <- c(11.1,10.6, 6.3, 8.8, 10.7, 11.2, 8.9, 12.2)
solar.radiation
```

11.1 10.6 6.3 8.8 10.7 11.2 8.9 12.2

In [94]:

```
cat("Mean:", mean(solar.radiation))
cat("\nMedian:", median(solar.radiation))
cat("\nRange:", range(solar.radiation))
cat("\nVariance:", var(solar.radiation))
```

```
Mean: 9.975
Median: 10.65
Range: 6.3 12.2
Variance: 3.525
```

In [96]:

```
# added 10 to each element
sr10 <- solar.radiation + 10
sr10
```

21.1 20.6 16.3 18.8 20.7 21.2 18.9 22.2

In [97]:

```
cat("Mean:", mean(sr10))
cat("\nMedian:", median(sr10))
cat("\nRange:", range(sr10))
cat("\nVariance:", var(sr10))
```

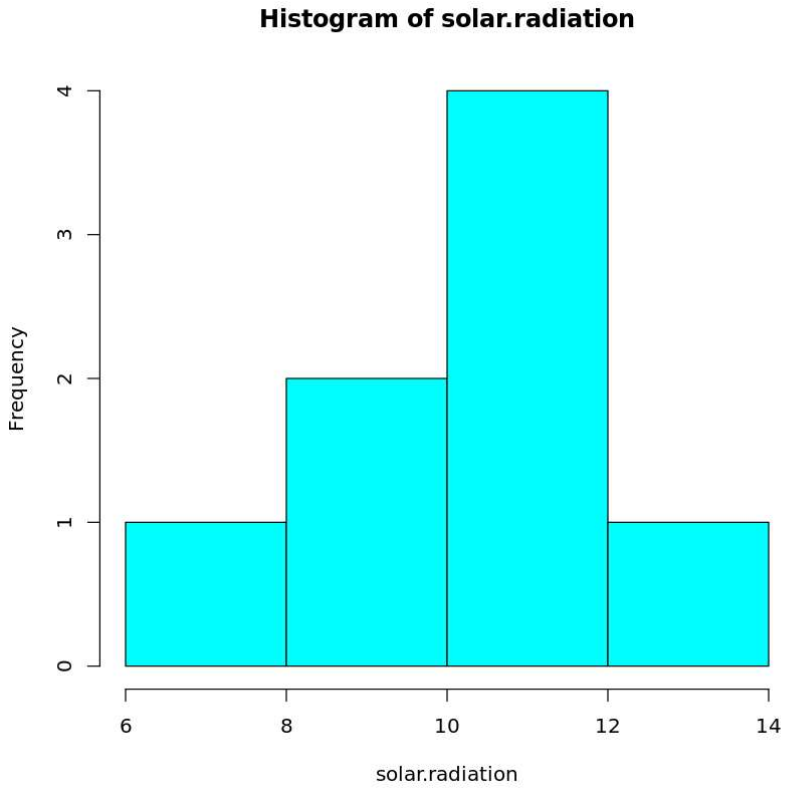
```
Mean: 19.975
Median: 20.65
Range: 16.3 22.2
Variance: 3.525
```

Answer:-

- Statistics of mean, median and range have changed and the variance value hasn't changed
- Mean, median and range have changed by +10

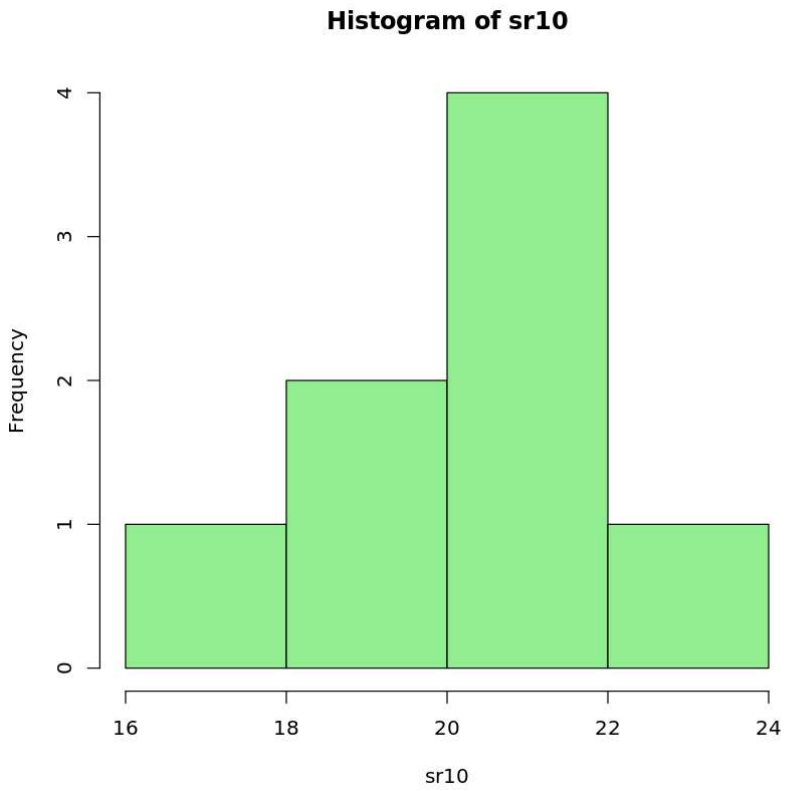
In [100]:

```
hist(solar.radiation, col ='cyan')
```



In [102]:

```
hist(sr10,col = 'light green')
```



4. Venn diagrams can be used to illustrate set unions and intersections. Draw Venn diagrams that correspond to and, or, not, and xor operations.

install.packages("BiocManager")

BiocManager::install("limma")

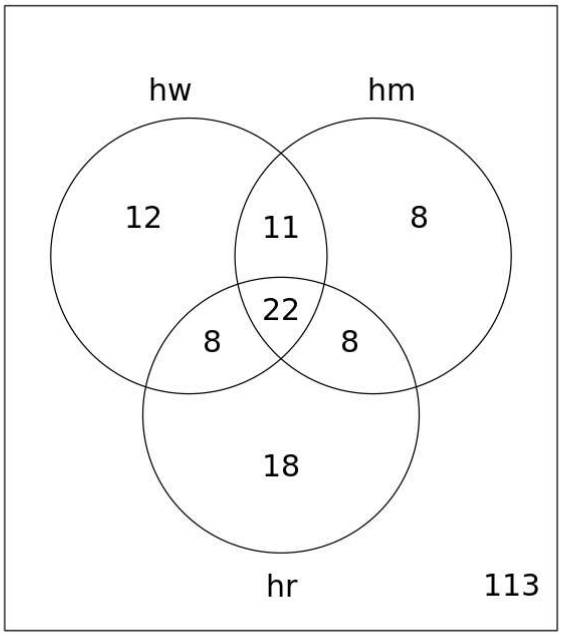
library(limma)

In [106]:

```
hsb2 <- read.csv("https://stats.idre.ucla.edu/wp-content/uploads/2016/02/hsb2-3.csv")
attach(hsb2)
hw <- (write >= 60)
hm <- (math >= 60)
hr <- (read >= 60)
c3 <- cbind(hw, hm, hr)
```

In [108]:

```
a <- vennCounts(c3)
vennDiagram(a)
```



5. Consider the built-in data frame cars.

a) Consult the help page to determine the number of observations in the dataset as well as the number of variables. Also, what are the names of the variables?
b) Find the mean stopping distance for all observations for which the speed was 20 miles per hour.

c) Construct a scatterplot relating stopping distance to speed. What kind of relationship do you observe?

In [190]:

```
df <- datasets::cars
head(df)
```

A data.frame: 6 × 2

	speed	dist
	<dbl>	<dbl>
1	4	2
2	4	10
3	7	4
4	7	22
5	8	16
6	9	10

In [166]:

```
# we can use dim() or nrow() to find the number of observations
nrow(df)  # here we have 50 observations in this dataframe
```

50

In [172]:

```
# finding the names of the column
colnames(df)
```

'speed' 'dist'

In [183]:

```
new_df <- df[df['speed'] == 20]
print(new_df)
```

```
 [1] 20 20 20 20 20 32 48 52 56 64
```

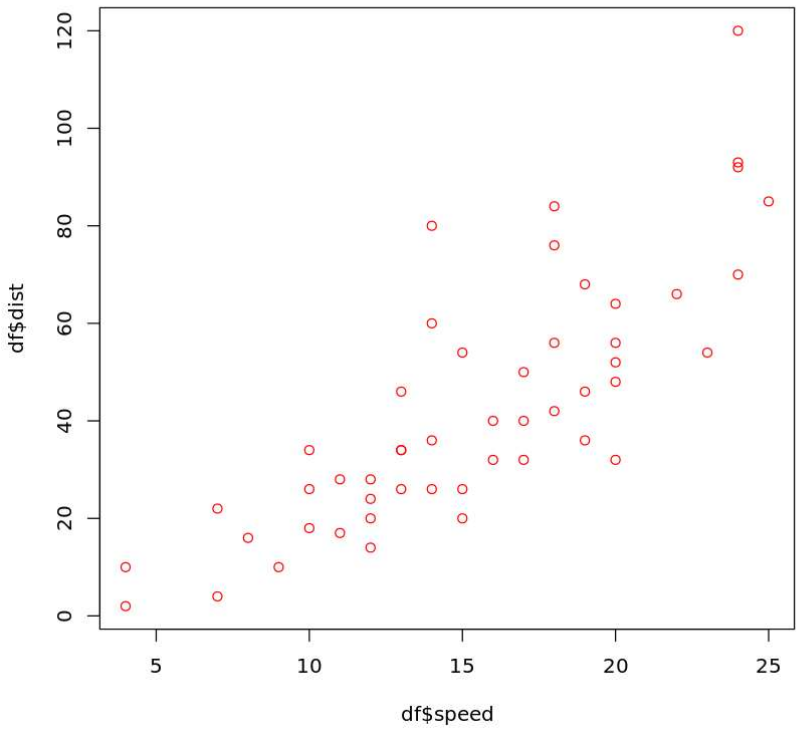
In [184]:

```
mean(new_df)  # mean stopping distance when speed is equal to 20
```

35.2

In [193]:

```
plot(df$speed, df$dist, col='red')
```



Conclusions that can be made from the plot:

- We observe that there is a linear realtionship between speed and stopping distance

Name:-Mithun G

USN:-19BTRCR006

LAB PROGRAM 6

1. What happens when the following code is run?
gender <- c("M" , "M" , "F" , "F" , "F")
whereF<- (gender == "F")
gender[whereF] <- "Female"

In [1]:

```
gender <- c("M" , "M" , "F" , "F" , "F" )
whereF <- (gender == "F" )
gender[whereF] <- "Female"
gender
```

'M' · 'M' · 'Female' · 'Female' · 'Female'

2. Construct the data frame char-num in the following way:
char <- c("2" , "1" , "0")
num<- 0:2
charnum<- data.frame(char, num);
Explore sampling distribution and central limit theorem in R

In [2]:

```
char <- c("2" , "1" , "0" )
num <- 0:2
charnum <- data.frame(char, num);
charnum
```

A data.frame:

3 × 2

char	num
<fct>	<int>
2	0
1	1
0	2

In [3]:

```
summary(charnum)
# since one of the variable is of character datatype, population mean and sample variance c
```

```
 char       num
 0:1   Min.   :0.0
 1:1   1st Qu.:0.5
 2:1   Median :1.0
       Mean   :1.0
       3rd Qu.:1.5
       Max.   :2.0
```

In [4]:

```
# sample variance
var(charnum[['num']])
```

1

3. Use the inbuilt data car and uses the possible graphical plots using ggplot2 graphical packages.

In [5]:

```
df <- datasets::mtcars
head(df)
```

A data.frame: 6 × 11

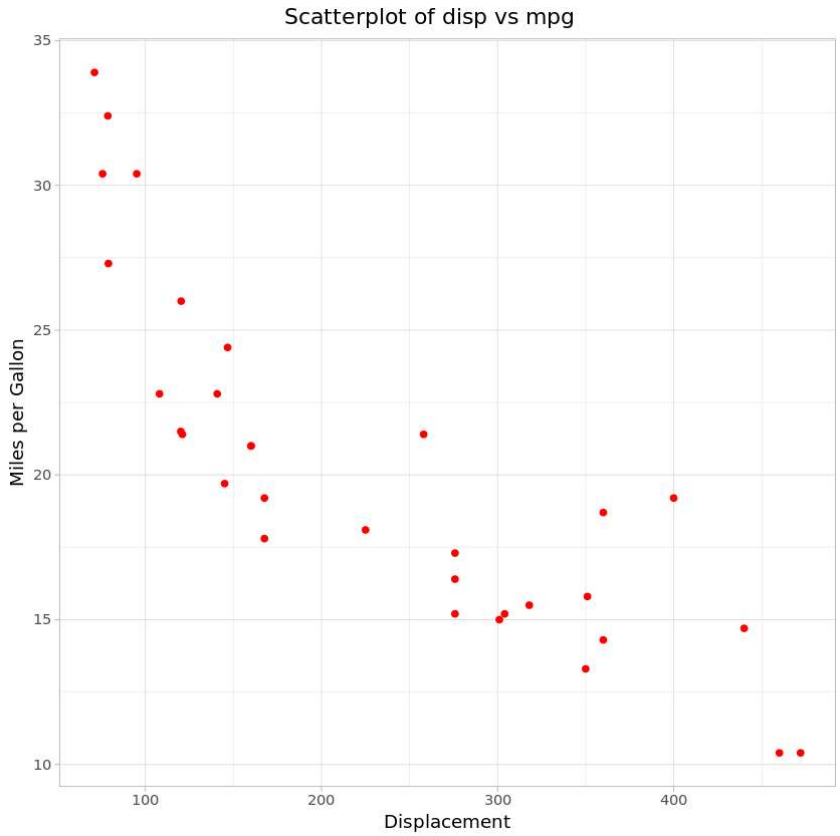
	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

install.packages("ggalt") install.packages('GGally') install.packages('ggridges')

```
library('tidyverse')
library('ggalt')
library('GGally')
library('ggridges')
```

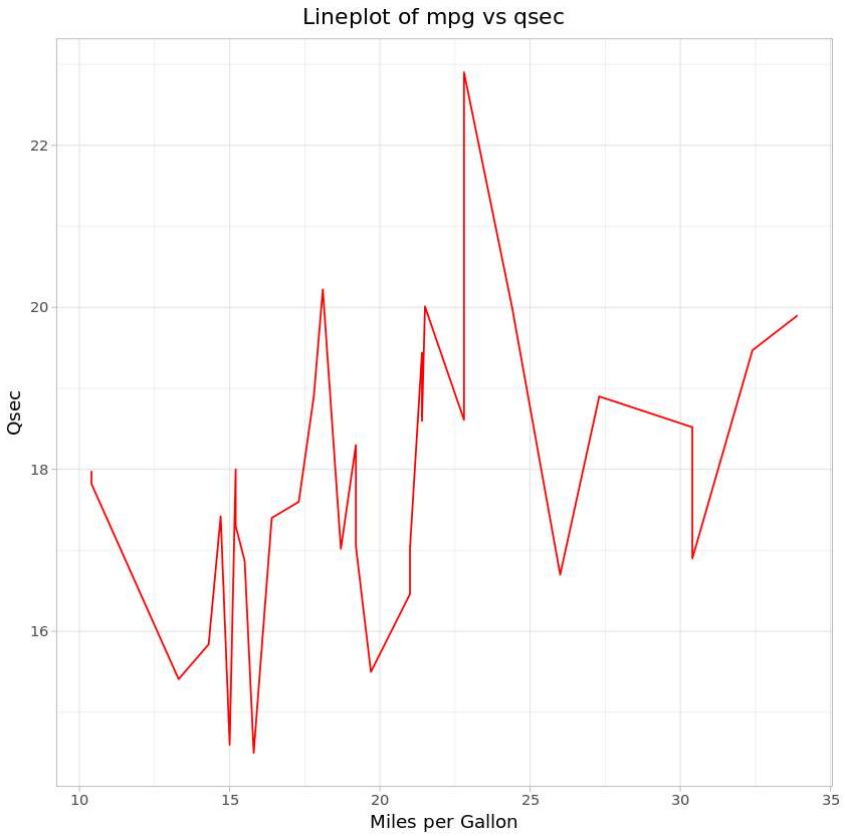
In [96]:

```
# scatterplot
ggplot(data=df) + geom_point(aes(x=disp, y=mpg), color = 'red') + theme_light() +
ggtitle("\t\t\t\tScatterplot of disp vs mpg") + xlab("Displacement") + ylab('Miles per Gall
```



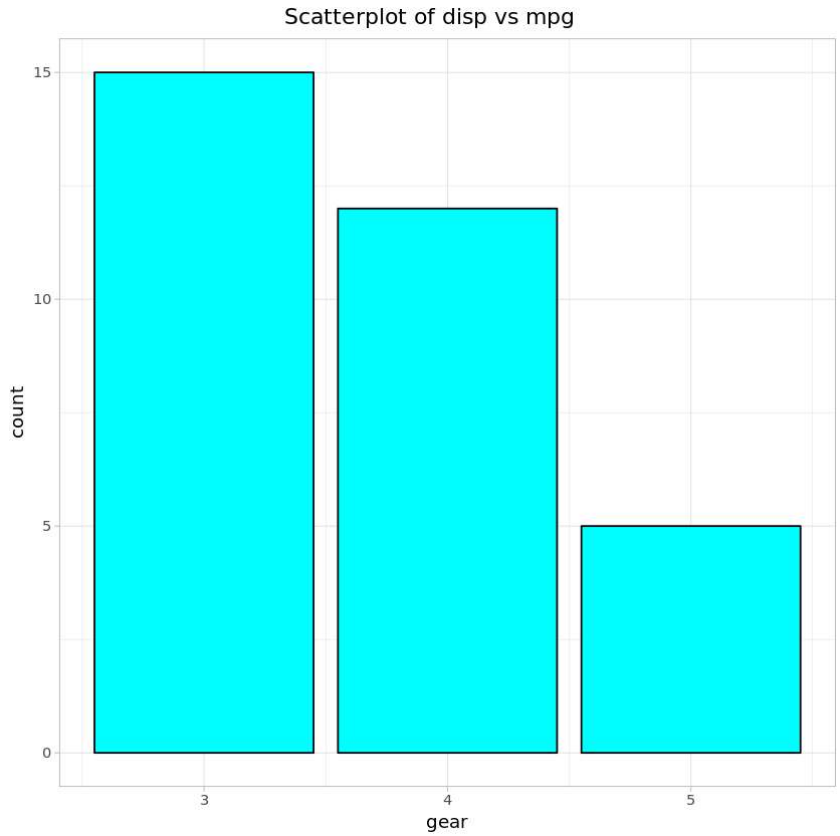
In [57]:

```
# Lineplot
ggplot(data=df) + geom_line(aes(x=mpg, y=qsec), color = 'red') + theme_light() +
ggtitle("\t\t\t\t   Lineplot of mpg vs qsec") + xlab("Miles per Gallon") + ylab('Qsec')
```



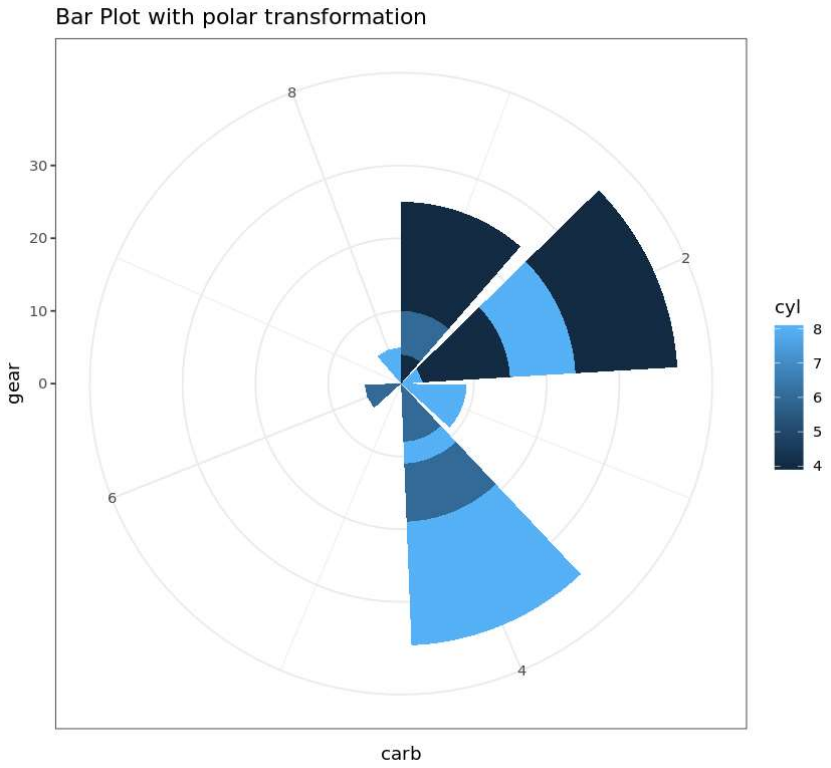
In [66]:

```
# Barplot -> univariate analysis
ggplot(data=df) + geom_bar(aes(gear), color = 'black', fill= 'cyan') + theme_light() +
ggtitle("\t\t\t\t Barplot of gear")
```



In [82]:

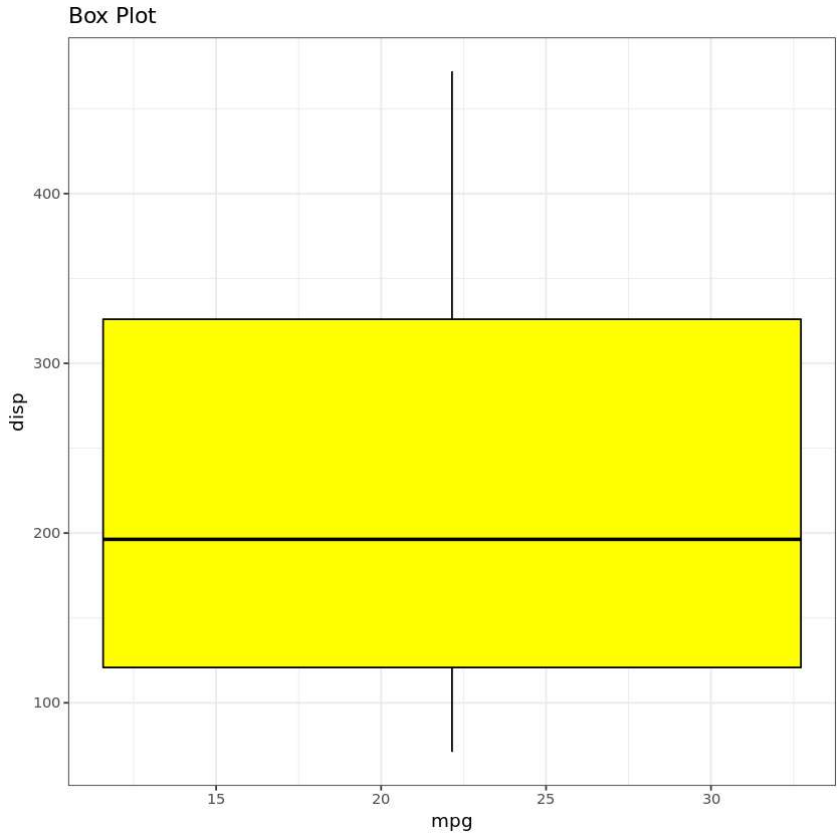
```
# polar transformation
ggplot(df) + geom_bar(aes(x = carb, y = gear, fill = cyl), stat = 'identity')
+ ggtitle("Bar Plot with polar transformation") + theme_bw() + coord_polar("x")
```



In [106]:

```
# boxplot
ggplot(df) + geom_boxplot(aes(x = mpg, y = disp), color='black', fill='yellow') + ggtitle("
```

```
Warning message:
“Continuous x aesthetic -- did you forget aes(group=...)?”
```



1. Modify the code to generate the Fibonacci sequence in the following ways.

a) Change the first two elements to 2 and 2.

b) Change the first two elements to 3 and 2.

c) Change the update rule from summing successive elements to taking differences of successive elements. For example, the third element is defined as the second element minus the first element, and so on.

d) Change the update rule so that each element is defined as the sum of the three preceding elements. Set the third element as 1 in order to start the process.

In [12]:

```
# Fibonacci function
fib <- function(n1,n2){
nterms = 7
count = 2
if(nterms <= 0) {
print("Plese enter a positive integer")
} else {
    if(nterms == 1) {
    print("Fibonacci sequence:")
    print(n1)
} else {
print("Fibonacci sequence:")
print(n1)
print(n2)
while(count < nterms) {
nth = n1 + n2
print(nth)
n1 = n2
n2 = nth
count = count + 1
}
}
}
}
```

In [13]:

```
fib(0,1)  # normal fibonnaci
```

```
[1] "Fibonacci sequence:"
[1] 0
[1] 1
[1] 1
[1] 2
[1] 3
[1] 5
[1] 8
```

In [14]:

```
fib(2,2) # changig first 2 elements to 2,2
```

```
[1] "Fibonacci sequence:"
[1] 2
[1] 2
[1] 4
[1] 6
[1] 10
[1] 16
[1] 26
```

In [15]:

```
fib(3,2) # changinf first 2 elements to 3,2
```

```
[1] "Fibonacci sequence:"
[1] 3
[1] 2
[1] 5
[1] 7
[1] 12
[1] 19
[1] 31
```

In [3]:

```
# Fibonacci function
fib_mod <- function(n1,n2){
nterms = 7
count = 2
if(nterms <= 0) {
print("Plese enter a positive integer")
} else {
    if(nterms == 1) {
    print("Fibonacci sequence:")
    print(n1)
} else {
print("Fibonacci sequence:")
print(n1)
print(n2)
while(count < nterms) {
nth = n2 - n1
print(nth)
n1 = n2
n2 = nth
count = count+1
}
}
}
}
fib_mod(0,1)
```

```
[1] "Fibonacci sequence:"
[1] 0
[1] 1
[1] 1
[1] 0
[1] -1
[1] -1
[1] 0
```

Name: Mithun G

USN:19BTRCR006

LAB PROGRAM 7

1. In each of the following, determine the final value of answer. Check your result by running the code in R.
a) answer <- 0 for (j in 1:5) answer <- answer + j
b) answer <- NULL for (j in 1:5) answer <- c(answer, j)
c) answer <- 0 for (j in 1:5) answer <- c(answer, j)
d) answer <- 1 for (j in 1:5) answer <- answer * j)

In [3]:

```
# a
answer <- 0
for(j in 1:5){
    answer<- (answer+j)
}
answer
```

15

In [2]:

```
# b
answer <- NULL
for(j in 1:5){
    answer<- c(answer,j)
}
answer
```

1 · 2 · 3 · 4 · 5

In [4]:

```
# c
answer <- 0
for(j in 1:5){
    answer<- c(answer,j)
}
answer
```

0 · 1 · 2 · 3 · 4 · 5

In [5]:

```
# d
answer <- 1
for(j in 1:5){
    answer<- c(answer*j)
}
answer
```

120

2. Does the Eratosthenes() function work properly if n is not an integer? Is an error message required in this case?

In [17]:

```
Eratosthenes <- function(n) {
if (n >= 2) {
     x = seq(2, n)
     prime_nums = c()
     for (i in seq(2, n)) {
         if (any(x == i)) {
             prime_nums = c(prime_nums, i)
             x = c(x[(x %% i) != 0], i)
         }
     }
     return(prime_nums)
 }
 else
 {
     stop("Input number should be at least 2.")
 }
 }
Eratosthenes(12)
Eratosthenes(10.5)
```

2 · 3 · 5 · 7 · 11

2 · 3 · 5 · 7

Answer: The function Eratosthenes() can take input even if it's not an integer, it just considers the whole number part of it

3. Use the idea of the Eratosthenes() function to prove that there are infinitely many primes. Hint: suppose all primes were less than m, and construct a larger value n that would not be eliminated by the sieve.

In [20]:

```
Eratosthenes(1) # we see that there should be minimum 2 to get the sieve
```

```
Error in Eratosthenes(1): Input number should be at least 2.
Traceback:

1. Eratosthenes(1)
2. stop("Input number should be at least 2.")   # at line 15 of file <text>
```

In [21]:

```
Eratosthenes(2000) # we see that this can go infintely many primes, hence proved
```

```
 2 ·  3 ·  5 ·  7 ·  11 ·  13 ·  17 ·  19 ·  23 ·  29 ·  31 ·  37 ·  41 ·  43 ·  47 ·  53 ·
 59 ·  61 ·  67 ·  71 ·  73 ·  79 ·  83 ·  89 ·  97 ·  101 ·  103 ·  107 ·  109 ·  113 ·
 127 ·  131 ·  137 ·  139 ·  149 ·  151 ·  157 ·  163 ·  167 ·  173 ·  179 ·  181 ·  191 ·
 193 ·  197 ·  199 ·  211 ·  223 ·  227 ·  229 ·  233 ·  239 ·  241 ·  251 ·  257 ·  263 ·
 269 ·  271 ·  277 ·  281 ·  283 ·  293 ·  307 ·  311 ·  313 ·  317 ·  331 ·  337 ·  347 ·
 349 ·  353 ·  359 ·  367 ·  373 ·  379 ·  383 ·  389 ·  397 ·  401 ·  409 ·  419 ·  421 ·
 431 ·  433 ·  439 ·  443 ·  449 ·  457 ·  461 ·  463 ·  467 ·  479 ·  487 ·  491 ·  499 ·
 503 ·  509 ·  521 ·  523 ·  541 ·  547 ·  557 ·  563 ·  569 ·  571 ·  577 ·  587 ·  593 ·
 599 ·  601 ·  607 ·  613 ·  617 ·  619 ·  631 ·  641 ·  643 ·  647 ·  653 ·  659 ·  661 ·
 673 ·  677 ·  683 ·  691 ·  701 ·  709 ·  719 ·  727 ·  733 ·  739 ·  743 ·  751 ·  757 ·
 761 ·  769 ·  773 ·  787 ·  797 ·  809 ·  811 ·  821 ·  823 ·  827 ·  829 ·  839 ·  853 ·
 857 ·  859 ·  863 ·  877 ·  881 ·  883 ·  887 ·  907 ·  911 ·  919 ·  929 ·  937 ·  941 ·
 947 ·  953 ·  967 ·  971 ·  977 ·  983 ·  991 ·  997 ·  1009 ·  1013 ·  1019 ·  1021 ·
 1031 ·  1033 ·  1039 ·  1049 ·  1051 ·  1061 ·  1063 ·  1069 ·  1087 ·  1091 ·  1093 ·
 1097 ·  1103 ·  1109 ·  1117 ·  1123 ·  1129 ·  1151 ·  1153 ·  1163 ·  1171 ·  1181 ·
 1187 ·  1193 ·  1201 ·  1213 ·  1217 ·  1223 ·  1229 ·  1231 ·  1237 ·  1249 ·  1259 ·
 1277 ·  1279 ·  1283 ·  1289 ·  1291 ·  1297 ·  1301 ·  1303 ·  1307 ·  1319 ·  1321 ·
 1327 ·  1361 ·  1367 ·  1373 ·  1381 ·  1399 ·  1409 ·  1423 ·  1427 ·  1429 ·  1433 ·
 1439 ·  1447 ·  1451 ·  1453 ·  1459 ·  1471 ·  1481 ·  1483 ·  1487 ·  1489 ·  1493 ·
 1499 ·  1511 ·  1523 ·  1531 ·  1543 ·  1549 ·  1553 ·  1559 ·  1567 ·  1571 ·  1579 ·
 1583 ·  1597 ·  1601 ·  1607 ·  1609 ·  1613 ·  1619 ·  1621 ·  1627 ·  1637 ·  1657 ·
 1663 ·  1667 ·  1669 ·  1693 ·  1697 ·  1699 ·  1709 ·  1721 ·  1723 ·  1733 ·  1741 ·
 1747 ·  1753 ·  1759 ·  1777 ·  1783 ·  1787 ·  1789 ·  1801 ·  1811 ·  1823 ·  1831 ·
 1847 ·  1861 ·  1867 ·  1871 ·  1873 ·  1877 ·  1879 ·  1889 ·  1901 ·  1907 ·  1913 ·
 1931 ·  1933 ·  1949 ·  1951 ·  1973 ·  1979 ·  1987 ·  1993 ·  1997 ·  1999
```

4. Write an R function called compound.interest() which computes this amount. Your function should have three arguments.

In [24]:

```
compound.interest <- function(price, rate, n)
{
    total_interest <- 0
    for (i in n){
          total_interest <- total_interest + (price*(1+(rate)^n)/100)
}
    cat("The compound interest calculated is:",total_interest)
}

compound.interest(2000,3,5)
```

```
The compound interest calculated is: 4880
```

5. Consider the inbuilt data set “cars”.
a) Find Correlation between possible variables and pairwise correlation.
b) Find regression line between appropriate variables.
c) Display the summary statistics and comment on the results.

In [26]:

```
df <- datasets::cars
head(df)
```

A data.frame: 6 × 2

	speed	dist
	<dbl>	<dbl>
1	4	2
2	4	10
3	7	4
4	7	22
5	8	16
6	9	10

In [28]:

```
# finding correaltion between possible variables and pairwise correaltion
cor(df$speed, df$dist)
```

0.80689490068921

In [29]:

```
# Find regression line between appropriate variables.
model <- lm(dist ~ speed, data=df)    # this gives the best fit line's parameters
print(model)
```

```
Call:
lm(formula = dist ~ speed, data = df)

Coefficients:
(Intercept)        speed
    -17.579        3.932
```

In [30]:

```
summary(model)
```

```
Call:
lm(formula = dist ~ speed, data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-29.069  -9.525  -2.272   9.215  43.201

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791     6.7584  -2.601   0.0123 *
speed         3.9324     0.4155   9.464 1.49e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared:  0.6511,    Adjusted R-squared:  0.6438
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

From the summary() we can see that:

- the Linear model we used has a R-squared error of 0.6438 *(R squared error ranges from 0 to 1, higher the value better is the fit of the line)*
- The linear model has F-statistic of 89.57 on 1

Name: Mithun G

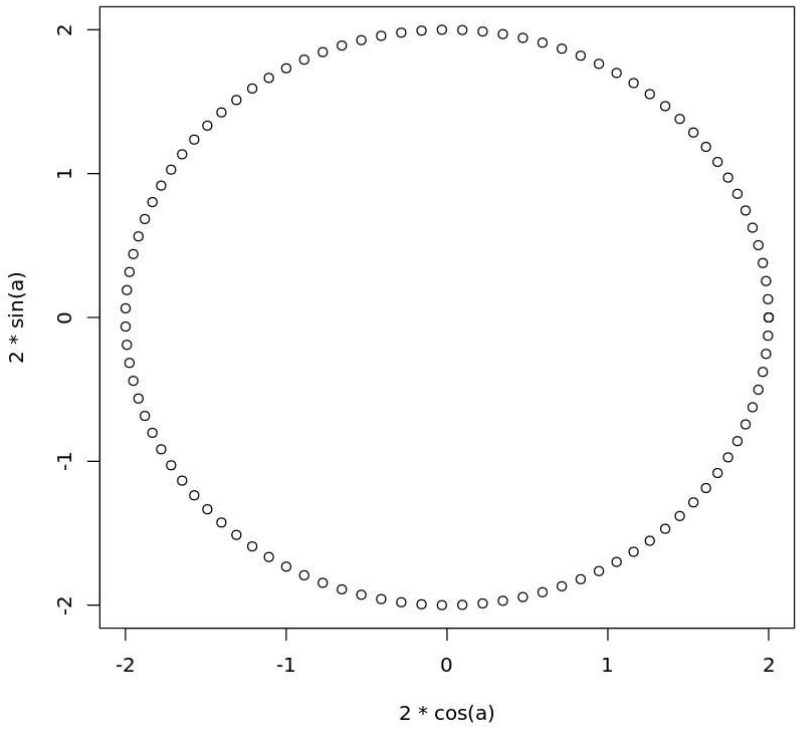
USN:19BTRCR006

LAB PROGRAM 8

1. Plot a circle by (x,y) points, where x = r · cos(a) and y = r · sin(a), with a the angle, from 0 to 2π, and r the radius.

In [4]:

```
a <- seq(0,2*pi, length.out = 100)
plot(x= 2*cos(a),y= 2*sin(a))   # here 2 is the radius
```

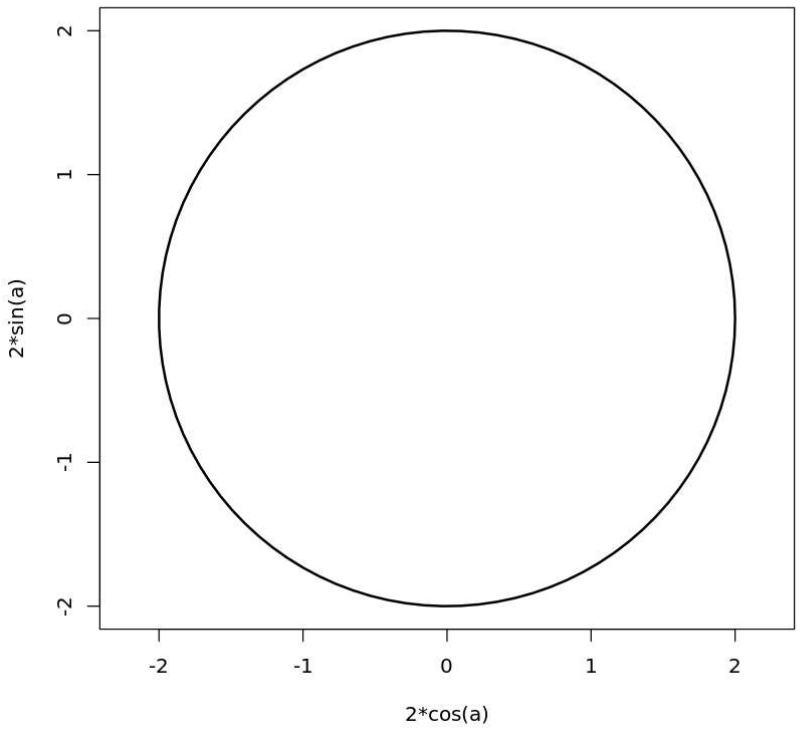


2. Using the same plot, add a number of graphical parameters that specify:

- **Rather than dots, the points should be connected by lines (type).**
- **The line should be twice as wide as the default (lwd).**
- **Give the appropriate x- and y-axes labels (xlab, ylab).**
- **Give the axes and axes annotations (axes).**
- **The graph has to be symmetrical, i.e. the x/y aspect ratio = 1 (asp).**
- **Add the legend, that describe the cure.**

In [5]:

```
plot(2*cos(a), 2*sin(a), type = "l", lwd = 2, xlab = "2*cos(a)", ylab="2*sin(a)", axes = TR
```

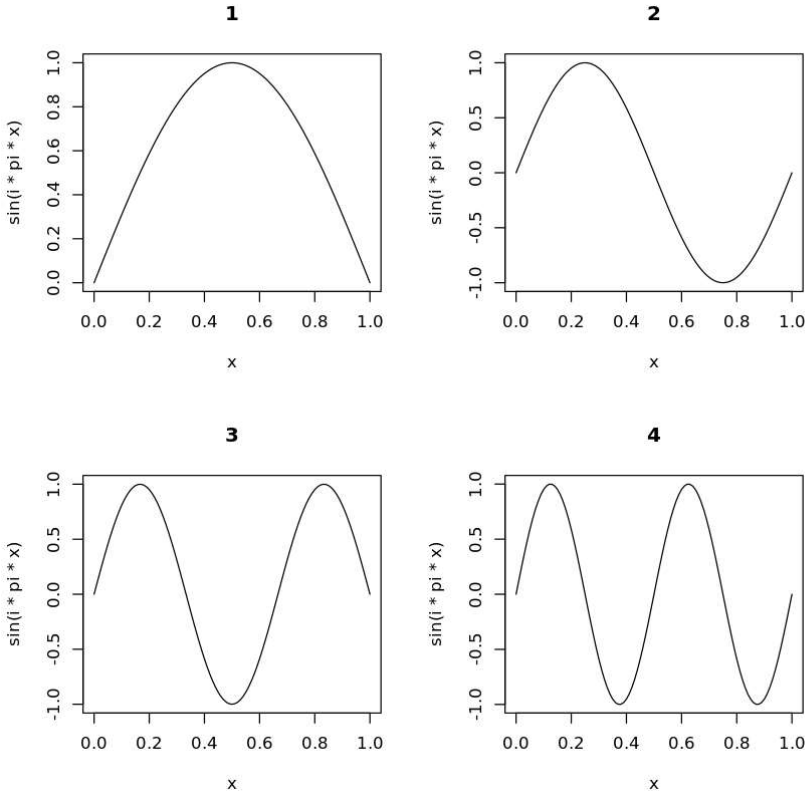


3. Explore for plotting multiple figures in a single window with partition as par() command as follows

```
par (mfrow=c(2,2))
for ( i in 1:4) curve(sin(ipix),0,1,main=i)
```

In [7]:

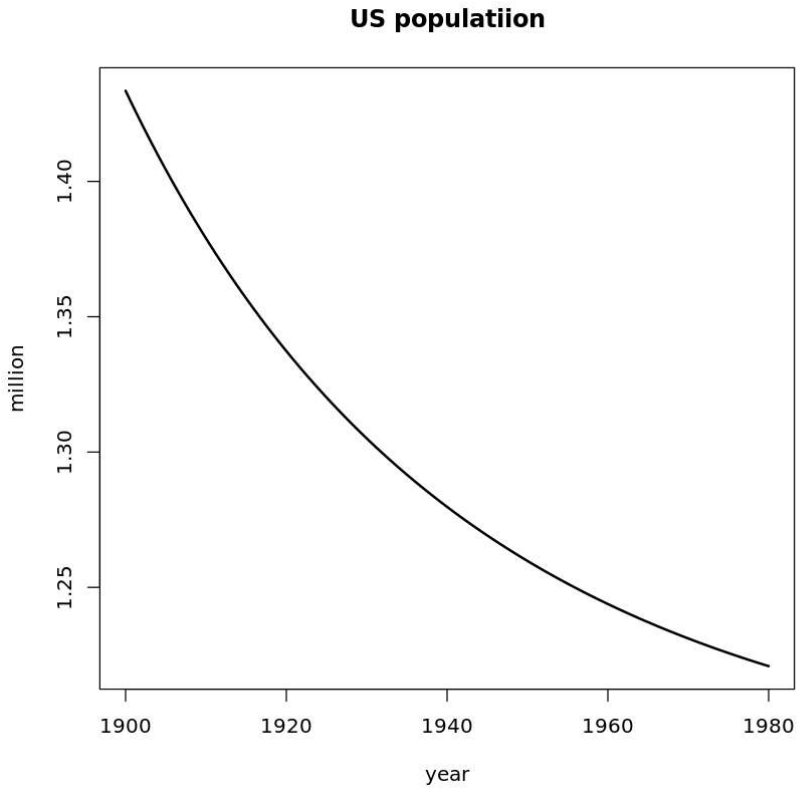
```
par(mfrow=c(2,2))
for(i in 1:4){
  curve(sin(i*pi*x),0,1,main=i)
}
```



4. For the US, the population density in 1900 (N0) was 76.1 million; the population growth can be described with parameter values: a=0.02 yr−1, K = 500 million of people.
Actual population values are:
1900 1910 1920 1930 1940 1950 1960 1970 1980
76.1 92.4 106.5 123.1 132.6 152.3 180.7 204.9 226.5
a) Plot the population density curve as a thick line, using the US parameter values.
b) Add the measured population values as points. Finish the graph with titles, labels etc.

In [8]:

```
k <- 500
no <- 76.1
a <- 0.02
curve(k/(1+((k-no)-no*exp(-a*(x-1900)))),1900,1980,main="US populatiion",xlab="year",ylab="
n <- matrix(ncol=2,data=c(seq(1900,1980,by=10), 76.1,92.4,106.5,123.1,132.6,152.3,180.7,204
points(n)
```

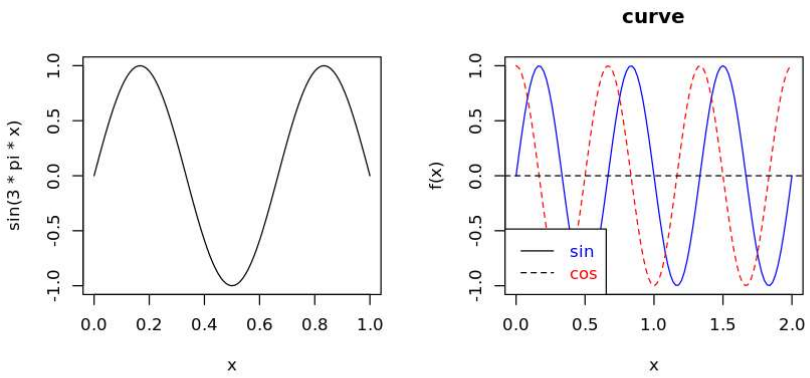


5. Plot curves for mathematical functions with following R-command ”curve”:

```
curve(sin(3pix))
curve(sin(3pix),from=0,to=2,col="blue", + xlab="x",ylab="f(x)",main="curve")
curve(cos(3pix),add=TRUE,col="red",lty=2) >abline(h=0,lty=2)
legend("bottomleft",c("sin","cos"),text.col=c("blue","red"),lty=1:2)
```

In [11]:

```
par(mfrow=c(2,2))
curve(sin(3*pi*x))
curve(sin(3*pi*x),from = 0,to=2,col="blue",xlab="x",ylab="f(x)",main="curve")

curve(cos(3*pi*x),add=TRUE,col="red",lty=2)
abline(h=0,lty=2)

legend("bottomleft",c("sin","cos"),text.col=c("blue","red"),lty=1:2)
```



Name: Mithun G

USN: 19BTRCR006

LAB PROGRAM 9

1. A famous inbuilt data set that is part of R is the ”iris” data set (Fisher, 1936). It gives measurements, in centimeters for sepal length and width and petal length and width, respectively, for 50 flowers of the species Iris setosa, Iris versicolor and Iris virginica. Have a look at the data:
a) What is the class of the data set? Why?
b) What are the dimensions of the data set? (number of rows, columns)
c) Produce a scatter plot of petal length against petal width; produce an informative title and labels of the two axes.
d) Repeat the same graph, using different symbol colors for the three species.
e) Add a legend to the graph. Copy-paste the result to a WORD document. If you do not have WORD, make a PDF file of the graph.
f) Create a box-and whisker plot for sepal length where the data values are split into species groups; use as template the first example in the ”boxplot” help file.
g) Now produce a similar box-and whisker plot for all four morphological measurements, arranged in two rows and two columns. First specify the graphical parameter that arranges the plots two by two.

In [1]:

```
df <- datasets::iris
head(df)
```

A data.frame: 6 × 5

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
	<dbl>	<dbl>	<dbl>	<dbl>	<fct>
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

In [3]:

```
# class of the dataset
class(df) # Because the dataset is stored in the form of rows and tables which is a datafra
```

'data.frame'

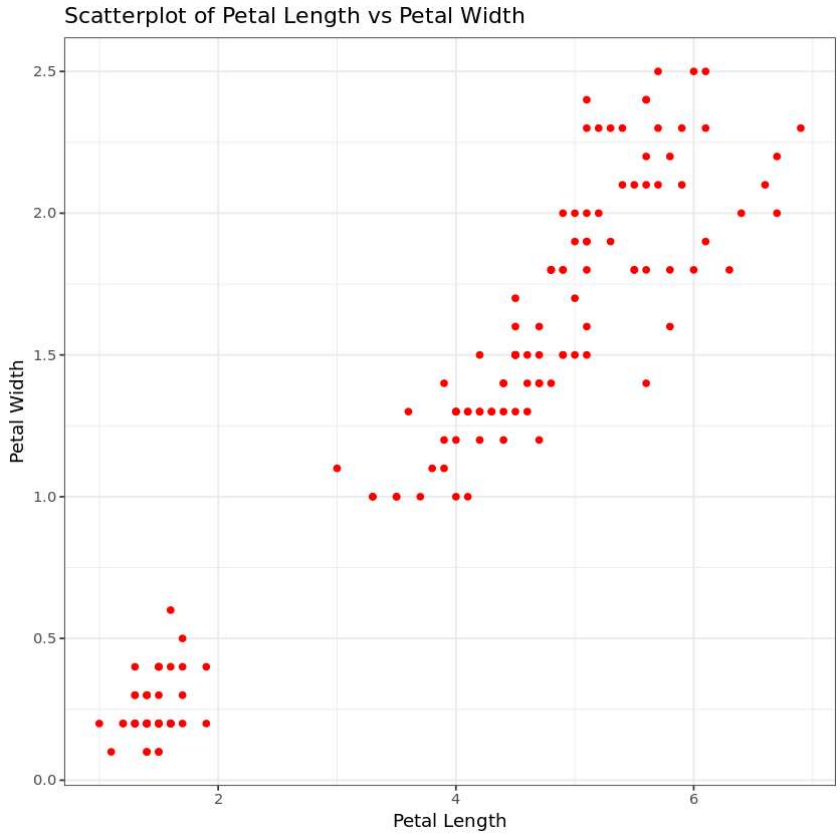
In [5]:

```
# dimensions of the dataset
dim(df)  # we have 150 rows and 5 columns
```

150 · 5

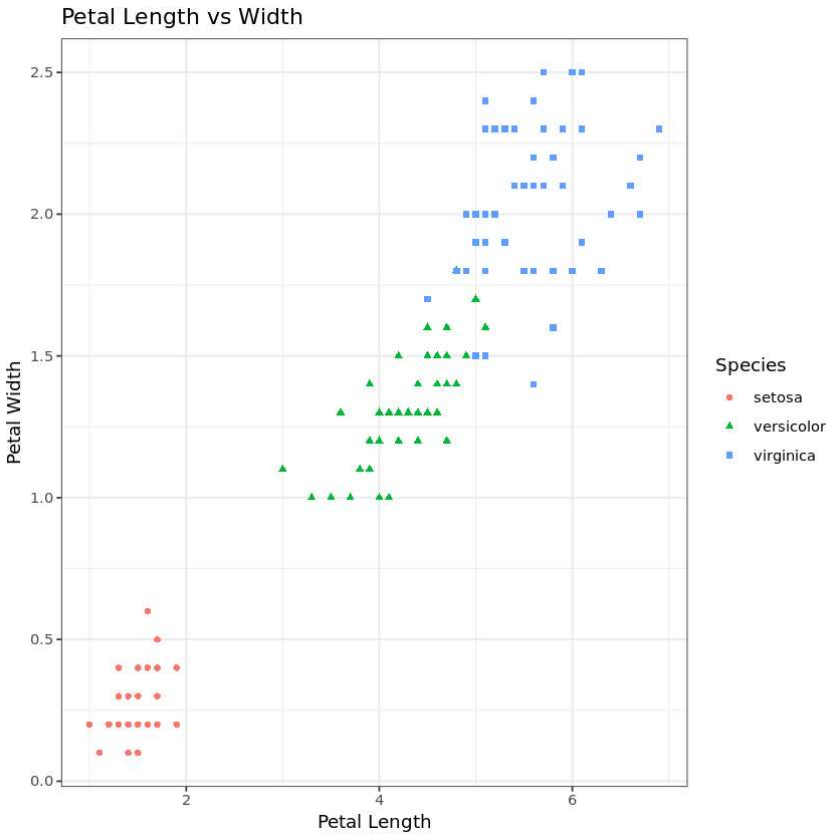
In [17]:

```
# scatter plot of petal length against petal width
library(tidyverse)
ggplot(df) + geom_point(aes(x=Petal.Length, y=Petal.Width), color='red') +theme_bw()+ xlab(
+ ggtitle("Scatterplot of Petal Length vs Petal Width")
```



In [28]:

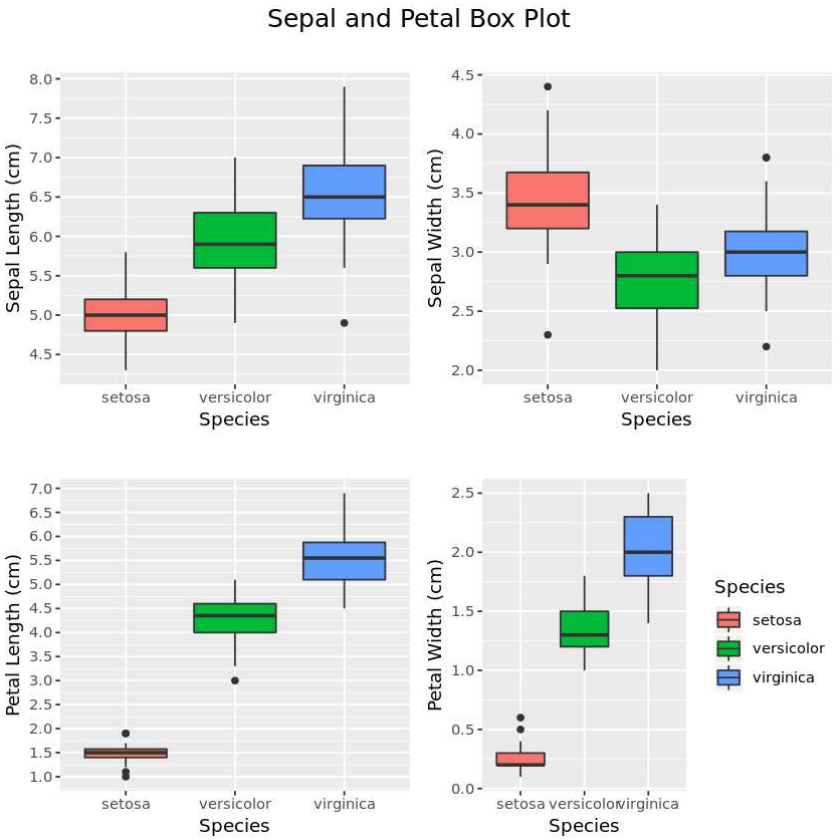
```
# using different symbol colors for the three species.
ggplot(data = df, aes(x = Petal.Length, y = Petal.Width))+ xlab("Petal Length")+ ylab("Peta
geom_point(aes(color = Species, shape=Species))+ ggtitle("Petal Length vs Width") + theme_b
```




```
library(grid)
library(gridExtra)
```

In [35]:

```
# boxplots


BpSl <- ggplot(df, aes(Species, Sepal.Length, fill=Species)) +
        geom_boxplot()+
        scale_y_continuous("Sepal Length (cm)", breaks= seq(0,30, by=.5))+
        theme(legend.position="none")



BpSw <-  ggplot(df, aes(Species, Sepal.Width, fill=Species)) +
          geom_boxplot()+
          scale_y_continuous("Sepal Width (cm)", breaks= seq(0,30, by=.5))+
          theme(legend.position="none")

BpPl <- ggplot(df, aes(Species, Petal.Length, fill=Species)) +
        geom_boxplot()+
        scale_y_continuous("Petal Length (cm)", breaks= seq(0,30, by=.5))+
        theme(legend.position="none")



BpPw <-  ggplot(df, aes(Species, Petal.Width, fill=Species)) +
        geom_boxplot()+
        scale_y_continuous("Petal Width (cm)", breaks= seq(0,30, by=.5))+
        labs(title = "Iris Box Plot", x = "Species")



# Plot all visualizations
grid.arrange(BpSl  + ggtitle(""),
             BpSw  + ggtitle(""),
             BpPl + ggtitle(""),
             BpPw + ggtitle(""),
             nrow = 2,
             top = textGrob("Sepal and Petal Box Plot", gp=gpar(fontsize=15)))
```



2. Write a script file that solves the following system of ODEs.

for initial values x=300,y=10 and parameter values: a=0.05, K=500, b=0.0002, g=0.8, e=0.03.
a) Make three plots, one for x and one for y as a function of time, and one plot expressing y as a function of x (this is called a phase-plane plot). Arrange these plots in 2 rows and 2 columns.
b) Now run the model with other initial values (x=200, y=50); add the (x,y) trajectories to the phase-plane plot

In [36]:

```
install.packages("deSolve")
library(deSolve)
```

```
Installing package into ‘/srv/rlibs’
(as ‘lib’ is unspecified)
```

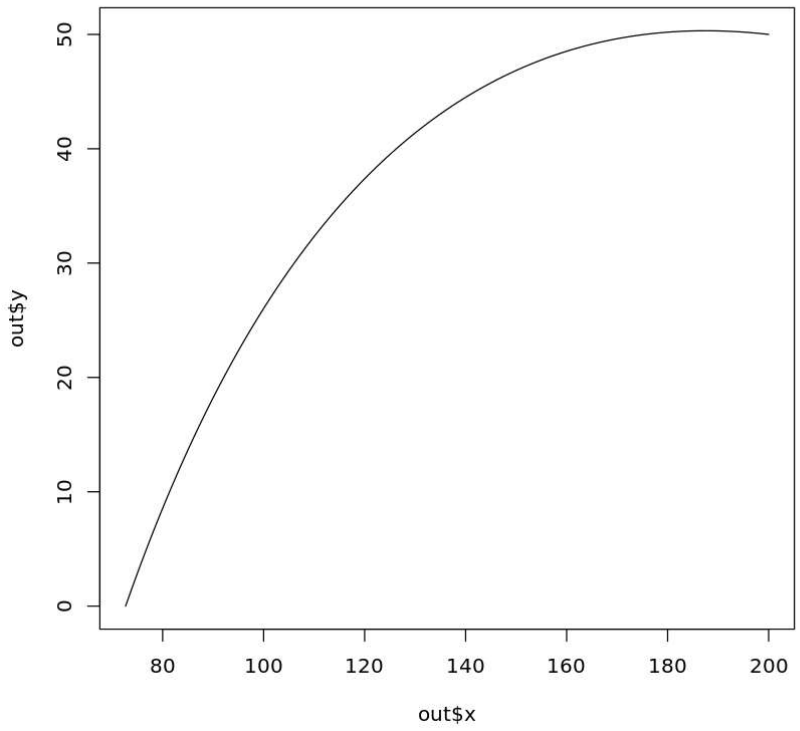
In [37]:

```
# a)
model <- function(time,VAR,pars){
  with(as.list(c(VAR,pars)), {
    dx <- a*x*(1-x/K)-b*x*y
    dy <- g*b*x*y - e*y
    return(list(c(dx,dy)))
  })
}
```

In [38]:

```
# b)
pars <- c(a=0,b=0.0002,K=500,g=0.8,e=0.03)
VAR <- c(x=200,y=50)
times <- seq(0,1000,1)
model(time,VAR,pars)
out <- as.data.frame(lsoda(VAR,times,model,pars))
plot(out$x,out$y,type="l")
```

1. -2 · 0.1



Name: Mithun G

USN: 19BTRCR006

LAB PROGRAM 10

1. Use R-function matrix to create the matrices called A and B:

a) Take the inverse of A and the transpose of A.
b) Multiply A with B.
c) Estimate the eigenvalues and eigenvectors of A.
d) For a matrix A, x is an eigenvector, and λ the eigenvalue of a matrix A, if A · x = λ · x. Test it!

In [1]:

```
A <- matrix(nrow=3, data=c(1,6,-2,2,4,1,3,1,-1))
B <- matrix(nrow = 3, data=c(1,2,-3,4,5,6,7,8,9))
print(A)
print(B)
```

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

In [13]:

```
# taking the inverse
print(solve(A))
cat("\n")
# finding the transpose
print(t(A))
```

```
            [,1]       [,2]       [,3]
[1,] -0.11111111  0.1111111 -0.2222222
[2,]  0.08888889  0.1111111  0.3777778
[3,]  0.31111111 -0.1111111 -0.1777778

     [,1] [,2] [,3]
[1,]    1    6   -2
[2,]    2    4    1
[3,]    3    1   -1
```

In [15]:

```
# Multiply with A and B
A %*% B
```

A matrix: 3 × 3
of type dbl

-4	32	50
11	50	83
3	-9	-15

In [18]:

```
# Eigen values
ev <- eigen(A)
values <- ev$values
values
```

6.36669562473516+0i · -1.18334781236758+2.38069708993942i · -1.18334781236758-2.38069708993942i

In [19]:

```
# Eigen vectors
vectors <- ev$vectors
vectors
```

A matrix: 3 × 3 of type cpl

-0.36275602+0i	-0.0725936-0.5240033i	-0.0725936+0.5240033i
-0.93146469+0i	-0.2632991+0.4856291i	-0.2632991-0.4856291i
-0.02795726+0i	0.6441961+0.0000000i	0.6441961+0.0000000i

In [21]:

```
# A.x
A %*% vectors
```

A matrix: 3 × 3 of type cpl

-2.3095572+0i	1.333397+0.447255i	1.333397-0.447255i
-5.9303521+0i	-0.844561-1.201503i	-0.844561+1.201503i
-0.1779954+0i	-0.762308+1.533636i	-0.762308-1.533636i

In [23]:

```
# λ.X
values * vectors
```

A matrix: 3 × 3 of type cpl

-2.30955718+0.00000000i	-0.462181-3.336170i	-0.462181+3.336170i
1.10224670-2.21753527i	-0.844561-1.201503i	1.467710-0.052167i
0.03308317+0.06655777i	-0.762308-1.533636i	-0.762308-1.533636i

Answer: From the last 2 cells we can see that A·x = λ·x

2. Create a matrix, called P:
a) What is the value of the largest eigenvalue (the so-called dominant eigenvalue) and the corresponding eigenvector?
b) Create a new matrix, T, which equals P, except for the first row, where the elements are 0.
c) Now estimate N= (I-T)-1, where I is the identity matrix.

In [24]:

```
P <- matrix(nrow = 4, data = c(0,0.9775,0,0,0.0043,0.9111,0.0736,0,0.1132,0,0.9534,0.0452,0
P
```

A matrix: 4 × 4 of type dbl

0.0000	0.0043	0.1132	0.0000
0.9775	0.9111	0.0000	0.0000
0.0000	0.0736	0.9534	0.0000
0.0000	0.0000	0.0452	0.9804

In [25]:

```
# Largest Eigen value
ev <- eigen(P)
values <- max(ev$values)
values
```

1.02544132553035

In [26]:

```
# Largest Eigen vector
vectors <- max(ev$vectors)
vectors
```

1

In [32]:

```
# creating the new matrix T
T <- P
T[1,] <- 0
T
```

A matrix: 4 × 4 of type dbl

0.0000	0.0000	0.0000	0.0000
0.9775	0.9111	0.0000	0.0000
0.0000	0.0736	0.9534	0.0000
0.0000	0.0000	0.0452	0.9804

In [39]:

```
# Estimating (I-T)-1
N <- solve((diag(4)-T)-1)
N
```

A matrix: 4 × 4 of type dbl

0.7334085	-0.2687986	-0.27249196	-0.1959587
5.0654149	5.2694115	-6.06133692	-4.3589244
2.2794650	2.5542928	6.03849018	-11.0896043
-8.3448799	-7.8237044	0.02284673	15.4485287

3. Find the root of the equation ex = 4x2 in the interval [0, 1]. And draw the function curve.

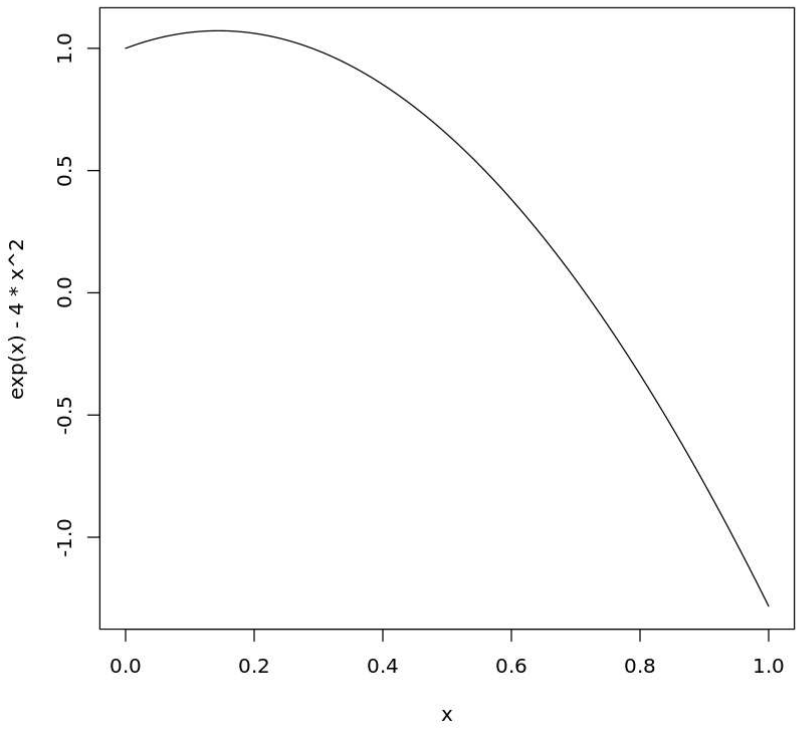
In [40]:

```
root <- uniroot(f=function(x) exp(x)-4*x^2,interval = c(0,1))
root
```

$root
0.714801396378604
$f.root
1.65946336081468e-05
$iter
5
$init.it
<NA>
$estim.prec
6.10351562503331e-05

In [41]:

```
curve(exp(x)-4*x^2,0,1)
```



4. Solve the equations 1000 = y ∗ (3 + x) ∗ (1 + y)4 for y and with x varying over the range from 1 to 100. Plot the root as a function of x.

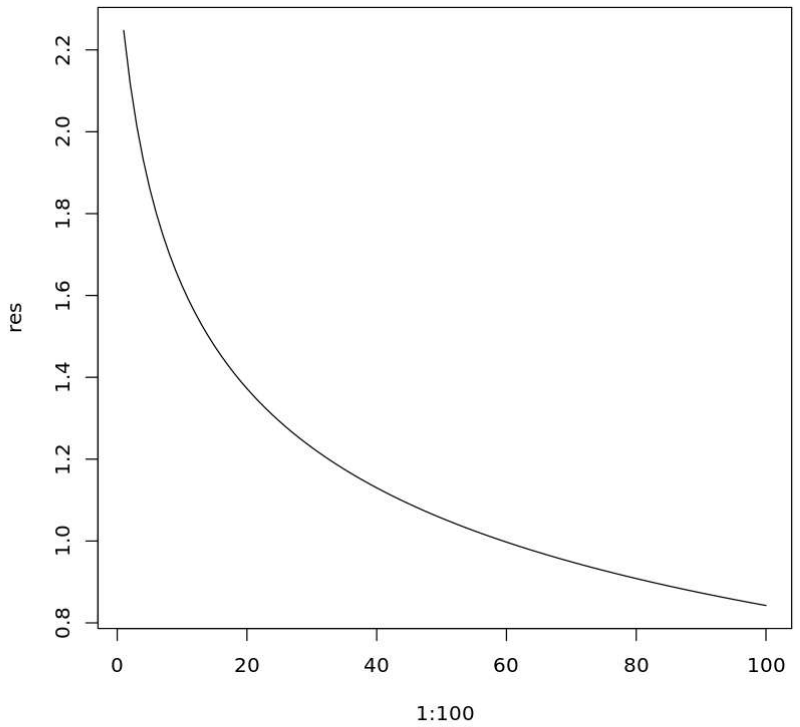
In [42]:

```
res <- vector()
for(x in 1:100){
    res[x]<-uniroot(f=function(y) y*(3+x)*(1+y)^4-1000,c(-1000,1000))
}
```

```
“number of items to replace is not a multiple of replacement length”
Warning message in res[x] <- uniroot(f = function(y) y * (3 + x) * (1 + y)
^4 - 1000, :
“number of items to replace is not a multiple of replacement length”
Warning message in res[x] <- uniroot(f = function(y) y * (3 + x) * (1 + y)
^4 - 1000, :
“number of items to replace is not a multiple of replacement length”
Warning message in res[x] <- uniroot(f = function(y) y * (3 + x) * (1 + y)
^4 - 1000, :
“number of items to replace is not a multiple of replacement length”
Warning message in res[x] <- uniroot(f = function(y) y * (3 + x) * (1 + y)
^4 - 1000, :
“number of items to replace is not a multiple of replacement length”
Warning message in res[x] <- uniroot(f = function(y) y * (3 + x) * (1 + y)
^4 - 1000, :
“number of items to replace is not a multiple of replacement length”
Warning message in res[x] <- uniroot(f = function(y) y * (3 + x) * (1 + y)
^4 - 1000, :
“number of items to replace is not a multiple of replacement length”
```

In [43]:

```
plot(1:100,res,type = "l")
```



Name: Mithun G
USN: 19BTRCR006

LAB PROGRAM 11

1. Create variables by taking input from user with readline(), cat(), and scan() commands and analyze the differences.

In [1]:

```
a <- as.integer(readline("Enter a number"))
print(a)
```

```
Enter a number7
[1] 7
```

In [2]:

```
b <- cat("Enter anything u please:",readline())
```

```
This is R
Enter anything u please: This is R
```

In [16]:

```
c <- scan("archive.csv", what = list("", "", ""))
c
```

1. 'Year,Punxsutawney' · 'Temperature,February' · '(Northeast),February' · '(Midwest),February' · '(Pennsylvania),March' · 'Average' · 'Average' · 'Average' · '1886,No' · 'Shadow,,,,,,,,' · '1889,No' · 'Shadow,,,,,,,,' · '1892,No' · 'Record,,,,,,,,' · '1895,No' · 'Record,35.04,22.2,33.5,26.6,38.03,25.3,36.9,27.8' · '1898,Full' · 'Record,25.5,18.1,22.2,20,37.63,29.3,38.4,34' · '1901,Full' · 'Record,31.46,20.1,23.6,21,41.58,37.1,43.9,38.8' · '1904,Full' · 'Shadow,26.94,15.2,22.2,18.1,45.12,31.4,47.2,36.9' · '1907,Full' · 'Shadow,33.01,18.4,31.2,22.1,43.92,32.7,47.1,38.4' · '1910,Full' · 'Shadow,33.66,21.4,37.5,28.5,44.02,28.4,42.9,32.8' · '1913,Full' · 'Shadow,29.52,14.5,26.3,18.7,40.75,29.6,39,31.7' · '1916,Full' · 'Shadow,30.09,17.8,29.1,23,38.35,31.2,42.6,35.4' · '1919,Full' · 'Shadow,33.69,19.6,31.5,23,40.28,32.7,42.9,36.7' · '1922,Full' · 'Shadow,29.57,16.2,29.1,22.1,38.3,27.9,40.1,34.3' · '1925,Full' · 'Shadow,37.67,20.5,36.3,25.8,39.65,25.9,36.3,30.2' · '1928,Full' · 'Shadow,26.92,21.7,26.2,24.7,43.56,36.2,47.5,41' · '1931,Full' · 'Shadow,36.7,25.7,41,32.6,37.36,28.1,36.3,30.8' · '1934,No' ·

2. Explore the string manipulation functions such as grep(), nchar(), paste(), sprint(), substr(), strsplit(),regexpr(), gregexpr(),

In [4]:

```
str <- "Riyuzaki"
str
```

'Riyuzaki'

In [5]:

```
grep("Pole",c("Equator","North Pole","South Pole"))
paste("North","Pole",sep=" ")
```

2 · 3

'North Pole'

In [6]:

```
nchar(str)
```

8

In [8]:

```
paste("hahaha", str)
```

'hahaha Riyuzaki'

In [9]:

```
sprintf("hahaha %s", str)
```

'hahaha Riyuzaki'

In [10]:

```
substring(str,1,4)
```

'Riyu'

In [11]:

```
strsplit("dead-beef", "-")
```

1. 'dead' · 'beef'

In [12]:

```
gregexpr("iss","Mississippi")
```

1. 2 · 5

3. Create variables by calling .csv, .xls, etc., files from system's storage and convert one format to another formats.

In [14]:

```
df <- read.csv('archive.csv')
head(df)
```

A data.frame: 6 × 10

	Year	Punxsutawney.Phil	February.Average.Temperature	February.Average.Temperature..Northe
	<fct>	<fct>	<dbl>	<d
1	1886	No Record	NA	
2	1887	Full Shadow	NA	
3	1888	Full Shadow	NA	
4	1889	No Record	NA	
5	1890	No Shadow	NA	
6	1891	No Record	NA	

In [15]:

```
tail(df)
```

A data.frame: 6 × 10

	Year	Punxsutawney.Phil	February.Average.Temperature	February.Average.Temperature..Nort
	<fct>	<fct>	<dbl>	
127	2012	Full Shadow	37.51	
128	2013	No Shadow	34.77	
129	2014	Full Shadow	32.13	
130	2015	Full Shadow	32.99	
131	2016	No Shadow	39.47	
132	1901-2000		33.82	

```
install.packages('openxlsx')
```

```
install.packages('rio')
```

In [38]:

```
library(openxlsx)
library("rio")
```

In [41]:

```
export(df, "archive.xlsx")
```

5. Write a program to set up socket connection with clients in R.

In [48]:

```
install.packages('svSocket')
```

```
Installing package into ‘/srv/rlibs’
(as ‘lib’ is unspecified)

also installing the dependency ‘svMisc’
```

In [*]:

```
library(svSocket)
startSocketServer()
# Start a second R process and run this code in it (the R client):
library(svSocket)
# Connect with the R socket server
con <- socketConnection(host = "localhost", port = 8888, blocking = FALSE)
L <- 10:20
L
evalServer(con, L) # L is not an the server, hence the error
evalServer(con, L, L) # Send it to the server
evalServer(con, L) # Now it is there
evalServer(con, L, L + 2)
L
evalServer(con, L)
```

```
Warning message:
“no DISPLAY variable so Tk is not available”
```

TRUE

10 · 11 · 12 · 13 · 14 · 15 · 16 · 17 · 18 · 19 · 20

Name: Mithun G

USN: 19BTRCR006

LAB PROGRAM 12

1. Create a table with the entries “S.No.”, “Name”, “Sub1Marks”, “Sub2Marks”, “Sub3Marks”, and “Sub4Marks” for 25 students of a class. And save it in a .csv format file.

In [125]:

```
name <- LETTERS[1:25]
table <- data.frame("S.no"=1:25,"name" = name, "Sub1Marks"=c(sample(0:100,5,replace=F)),"Su
table[0:5,]
```

A data.frame: 5 × 6

	S.no	name	Sub1Marks	Sub2Marks	Sub3Marks	Sub4Marks
	<int>	<fct>	<int>	<int>	<int>	<int>
1	1	A	72	91	45	6
2	2	B	99	97	68	19
3	3	C	2	2	57	15
4	4	D	58	38	88	72
5	5	E	0	55	100	74

In [23]:

```
write.csv(table,file="Students_lab.csv",row.names = T)
getwd()
```

'/home/jovyan'

2. Apply statistical commands such as summary(), str(), Names(), Rownames(), Columnnames(), and Dimensions() for the data of file and observe the outcomes.

In [24]:

```
# summary
summary(table)
```

```
      S.no          name      Sub1Marks       Sub2Marks       Sub3Marks    
 Min.   : 1   A       : 1   Min.   :27.0   Min.   :16.0   Min.   : 1  
 1st Qu.: 7   B       : 1   1st Qu.:28.0   1st Qu.:23.0   1st Qu.:14  
 Median :13   C       : 1   Median :48.0   Median :45.0   Median :25  
 Mean   :13   D       : 1   Mean   :47.4   Mean   :52.8   Mean   :21  
 3rd Qu.:19   E       : 1   3rd Qu.:57.0   3rd Qu.:83.0   3rd Qu.:31  
 Max.   :25   F       : 1   Max.   :77.0   Max.   :97.0   Max.   :34  
              (Other):19                                             
   Sub4Marks    
 Min.   :42.0  
 1st Qu.:68.0  
 Median :73.0  
 Mean   :70.8  
 3rd Qu.:81.0  
 Max.   :90.0  
```

In [26]:

```
#str()
str(table)
```

```
'data.frame':   25 obs. of  6 variables:
 $ S.no     : int  1 2 3 4 5 6 7 8 9 10 ...
 $ name     : Factor w/ 25 levels "A","B","C","D",..: 1 2 3 4 5 6 7 8 9 10
...
 $ Sub1Marks: int  28 77 57 48 27 28 77 57 48 27 ...
 $ Sub2Marks: int  97 16 83 23 45 97 16 83 23 45 ...
 $ Sub3Marks: int  34 1 25 14 31 34 1 25 14 31 ...
 $ Sub4Marks: int  90 68 73 42 81 90 68 73 42 81 ...
```

In [28]:

```
# names
names(table)
```

'S.no' · 'name' · 'Sub1Marks' · 'Sub2Marks' · 'Sub3Marks' · 'Sub4Marks'

In [29]:

```
# rownames
rownames(table)
```

'1' · '2' · '3' · '4' · '5' · '6' · '7' · '8' · '9' · '10' · '11' · '12' · '13' · '14' · '15' · '16' · '17' · '18' · '19' · '20' · '21' · '22' · '23' · '24' · '25'

In [33]:

```
# column names
colnames(table)
```

'S.no' · 'name' · 'Sub1Marks' · 'Sub2Marks' · 'Sub3Marks' · 'Sub4Marks'

In [34]:

```
# dimensions
dim(table)
```

25 · 6

3. Find log for each value in the data set with log() command and perform the summary() command.

In [42]:

```
log(table$Sub1Marks)
log(table$Sub2Marks)
log(table$Sub3Marks)
log(table$Sub4Marks)
```

3.3322045101752 · 4.34380542185368 · 4.04305126783455 · 3.87120101090789 · 3.29583686600433 · 3.3322045101752 · 4.34380542185368 · 4.04305126783455 · 3.87120101090789 · 3.29583686600433 · 3.3322045101752 · 4.34380542185368 · 4.04305126783455 · 3.87120101090789 · 3.29583686600433 · 3.3322045101752 · 4.34380542185368 · 4.04305126783455 · 3.87120101090789 · 3.29583686600433 · 3.3322045101752 · 4.34380542185368 · 4.04305126783455 · 3.87120101090789 · 3.29583686600433

4.57471097850338 · 2.77258872223978 · 4.4188406077966 · 3.13549421592915 · 3.80666248977032 · 4.57471097850338 · 2.77258872223978 · 4.4188406077966 · 3.13549421592915 · 3.80666248977032 · 4.57471097850338 · 2.77258872223978 · 4.4188406077966 · 3.13549421592915 · 3.80666248977032 · 4.57471097850338 · 2.77258872223978 · 4.4188406077966 · 3.13549421592915 · 3.80666248977032

3.52636052461616 · 0 · 3.2188758248682 · 2.63905732961526 · 3.43398720448515 · 3.52636052461616 · 0 · 3.2188758248682 · 2.63905732961526 · 3.43398720448515 · 3.52636052461616 · 0 · 3.2188758248682 · 2.63905732961526 · 3.43398720448515 · 3.52636052461616 · 0 · 3.2188758248682 · 2.63905732961526 · 3.43398720448515 · 3.52636052461616 · 0 · 3.2188758248682 · 2.63905732961526 · 3.43398720448515

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  3.738   4.220   4.290   4.228   4.394   4.500 
```

In [44]:

```
summary(log(table$Sub1Marks), log(table$Sub2Marks),log(table$Sub2Marks),log(table$Sub3Marks
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  3.296   3.332   3.871   3.777   4.043   4.344 
```

4. Apply the quantile() command and set the 4-quantile, 5-quantile, and 9-quantile values on the above data set.

In [59]:

```
quantile(table$Sub1Marks, type=4)
quantile(table$Sub1Marks, type=5)
quantile(table$Sub1Marks, type=9)
```

0%: 27 **25%:** 28 **50%:** 48 **75%:** 57 **100%:** 77

0%: 27 **25%:** 28 **50%:** 48 **75%:** 57 **100%:** 77

0%: 27 **25%:** 28 **50%:** 48 **75%:** 57 **100%:** 77

5. Perform the cumulative operation on different variables of data set with Cumsum(), Cummax(), Cummin(), and Cumprod() commands.

In [73]:

```
cumsum(table$Sub1Marks)
cummax(table$Sub1Marks)
cummin(table[['Sub1Marks']])
cumprod(table[['Sub1Marks']])
```

28 · 105 · 162 · 210 · 237 · 265 · 342 · 399 · 447 · 474 · 502 · 579 · 636 · 684 · 711 · 739 · 816 · 873 · 921 · 948 · 976 · 1053 · 1110 · 1158 · 1185

28 · 77 · 77 · 77 · 77 · 77 · 77 · 77 · 77 · 77 · 77 · 77 · 77 · 77 · 77 · 77 · 77 · 77 · 77 · 77 · 77 · 77 · 77 · 77 · 77 · 77

28 · 28 · 28 · 28 · 27 · 27 · 27 · 27 · 27 · 27 · 27 · 27 · 27 · 27 · 27 · 27 · 27 · 27 · 27 · 27 · 27 · 27 · 27 · 27 · 27

28 · 2156 · 122892 · 5898816 · 159268032 · 4459504896 · 343381876992 · 19572766988544 · 939492815450112 · 25366306017153024 · 710256568480284672 · 5468975572981919744 · 3.11731607905997e+21 · 1.49631171794879e+23 · 4.04004163846172e+24 · 1.13121165876928e+26 · 8.71032977252347e+27 · 4.96488797033838e+29 · 2.38314622576242e+31 · 6.43449480955854e+32 · 1.80165854667639e+34 · 1.38727708094082e+36 · 7.90747936136268e+37 · 3.79559009345408e+39 · 1.0248093252326e+41

6. Perform special summary commands for different rows and columns such as rowmeans(), rowsums(), colmeans(), and colsums() commands. Also observe the difference with apply() command.

In [103]:

```
Sub1marks <- table$Sub1Marks
Sub2marks <- table$Sub2Marks
Sub3marks <- table$Sub3Marks
Sub4marks <- table$Sub4Marks
```

In [104]:

```
x <- cbind(Sub1marks,Sub2marks,Sub3marks,Sub4marks)
rowMeans(x, na.rm = T)
group<-c(sample(0:25,1))
rowSums(x,na.rm = T)
colMeans(x,na.rm = T)
colSums(x, na.rm = T)
apply(x,2,sum)
```

62.25 · 40.5 · 59.5 · 31.75 · 46 · 62.25 · 40.5 · 59.5 · 31.75 · 46 · 62.25 · 40.5 · 59.5 · 31.75 · 46 · 62.25 · 40.5 · 59.5 · 31.75 · 46 · 62.25 · 40.5 · 59.5 · 31.75 · 46

249 · 162 · 238 · 127 · 184 · 249 · 162 · 238 · 127 · 184 · 249 · 162 · 238 · 127 · 184 · 249 · 162 · 238 · 127 · 184 · 249 · 162 · 238 · 127 · 184

Sub1marks: 47.4 **Sub2marks:** 52.8 **Sub3marks:** 21 **Sub4marks:** 70.8

Sub1marks: 1185 **Sub2marks:** 1320 **Sub3marks:** 525 **Sub4marks:** 1770

Sub1marks: 1185 **Sub2marks:** 1320 **Sub3marks:** 525 **Sub4marks:** 1770

7. Explore the descriptive statistics in R for Matrix object.

In [80]:

```
mat <- matrix(1:9, 3)
mat
```

A matrix:
3 × 3 of
type int

1	4	7
2	5	8
3	6	9

In [81]:

```
class(mat)
```

'matrix'

In [106]:

```
mean(mat)
median(mat)
sd(mat)
var <- sd(mat)*sd(mat)
var
```

5

5

2.73861278752583

7.5

8. Explore the descriptive statistics in R for lists.

In [111]:

```
lit <- list(1:9, c(10:17))
print(lit)
```

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

[[2]]
[1] 10 11 12 13 14 15 16 17
```

In [116]:

```
summary(lit)
```

```
     Length Class  Mode
[1,] 9      -none- numeric
[2,] 8      -none- numeric
```

In [121]:

```
mean(lit[[1]])
median(lit[[1]])
sd(lit[[1]])
var(lit[[1]])
```

5

5

2.73861278752583

7.5