DSEU OKHLA - PHASE II

(Formley - DITE)

DSEU

Crafting Excellence

LAB FILE

Department Of Computer Science & Engineering

B .Tech (CSE) 2024-26 4th Sem

PROBABILITY & STATISTICS BT-CS-BS-401

R Programming

Submitted By: **Manish Prasad**

Roll No: **41523056**

Branch: CSE

Submitted to:

Mr. P. Saravanan

Assistant Professor

CSE Department

Table of Content

EXPERIMENT-1

Aim -

To Study History and Installing a R 4.3.2 Programing Language .
Also,Creating Vectors, Sequence and Series in R

Theory-

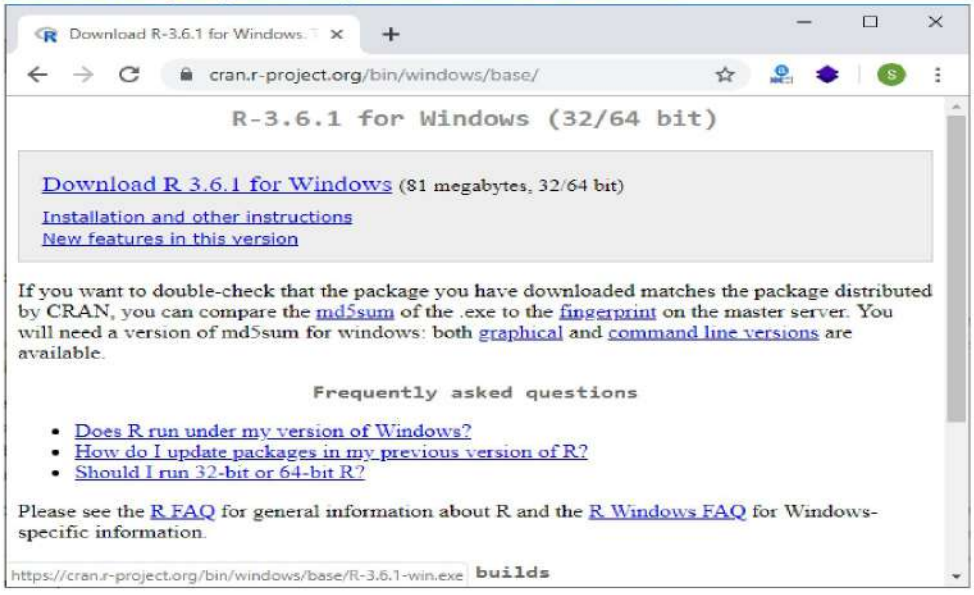
R is an open-source programming language that is widely used as a statistical software and data analysis tool. R generally comes with the Command-line interface. It was designed by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently being developed by the R Development Core Team.

It is a great resource for data analysis, data visualization, data science and machine learning
It is easy to draw graphs in R, like pie charts, histograms, box plot, scatter plot, etc.
It works on different platforms (Windows, Mac, Linux)
It is open-source and free and has many packages
It has a large community support

Installation-

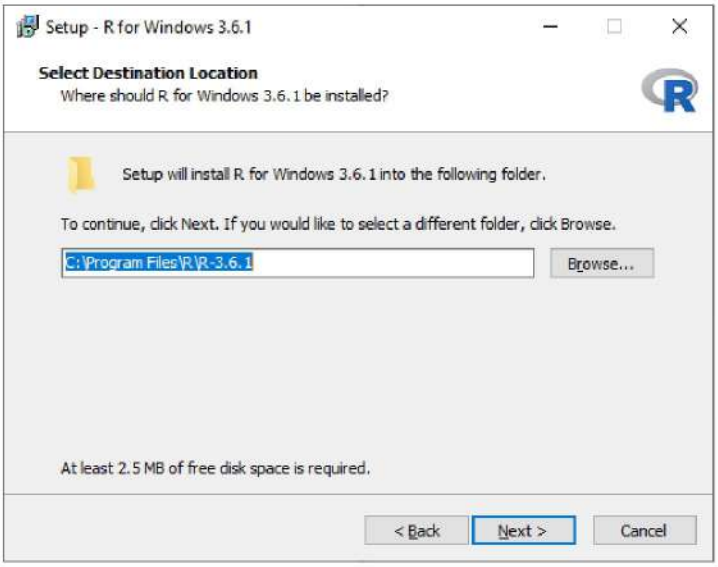
There are following steps used to install the R in Windows:

Step 1: First, We have to download the R setup from https://cloud.r-project.org/bin/windows/base/.

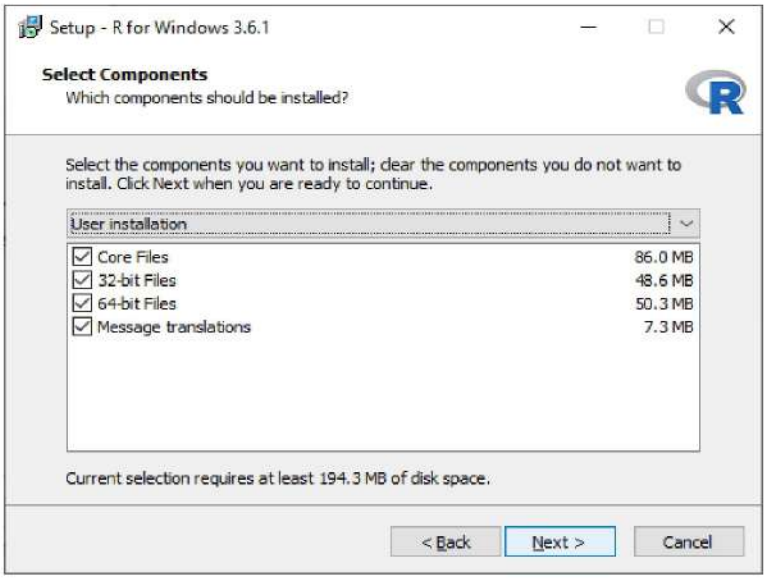


Step 2: When we click on Download R 4.3.2 for windows, our downloading will be started of R setup. Once the downloading is finished, we have to run the setup of R in the following way:

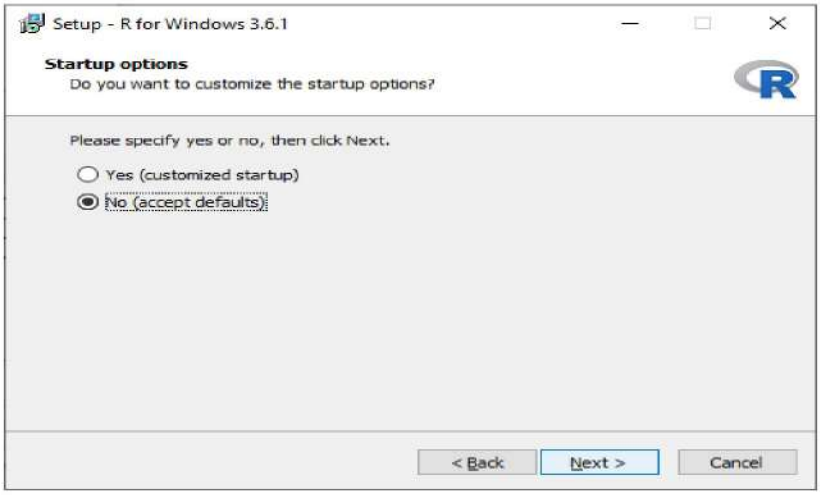
1) Select the path where we want to download the R and proceed to Next.



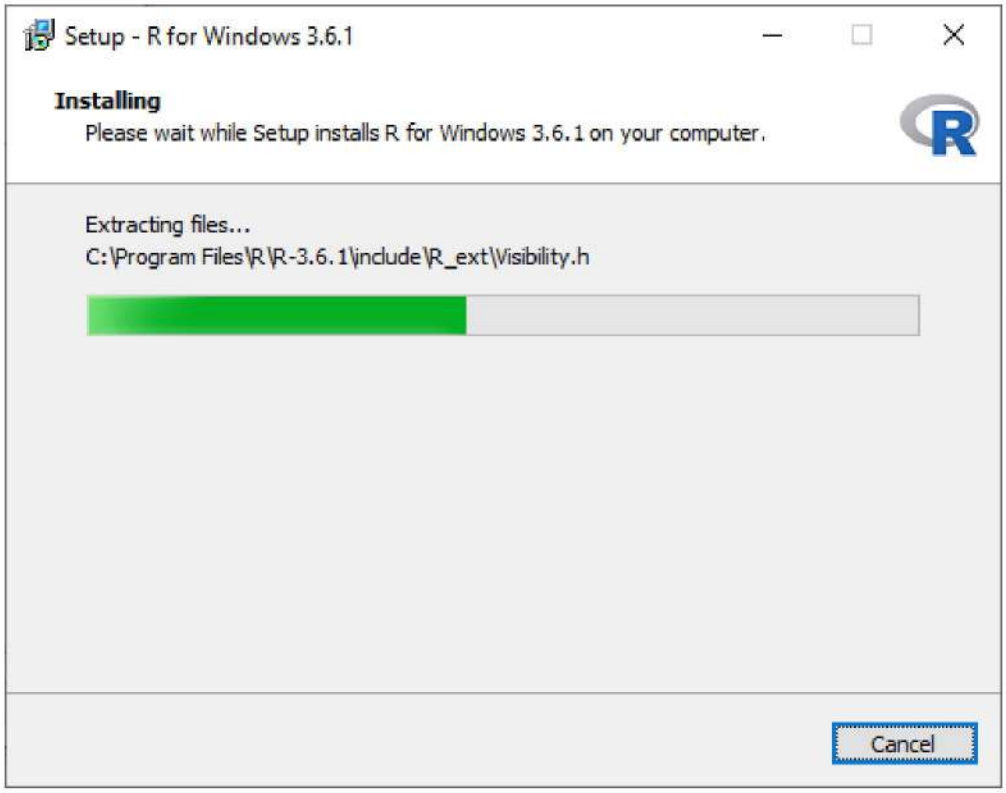
2) Select all components which we want to install, and then we will proceed to Next.



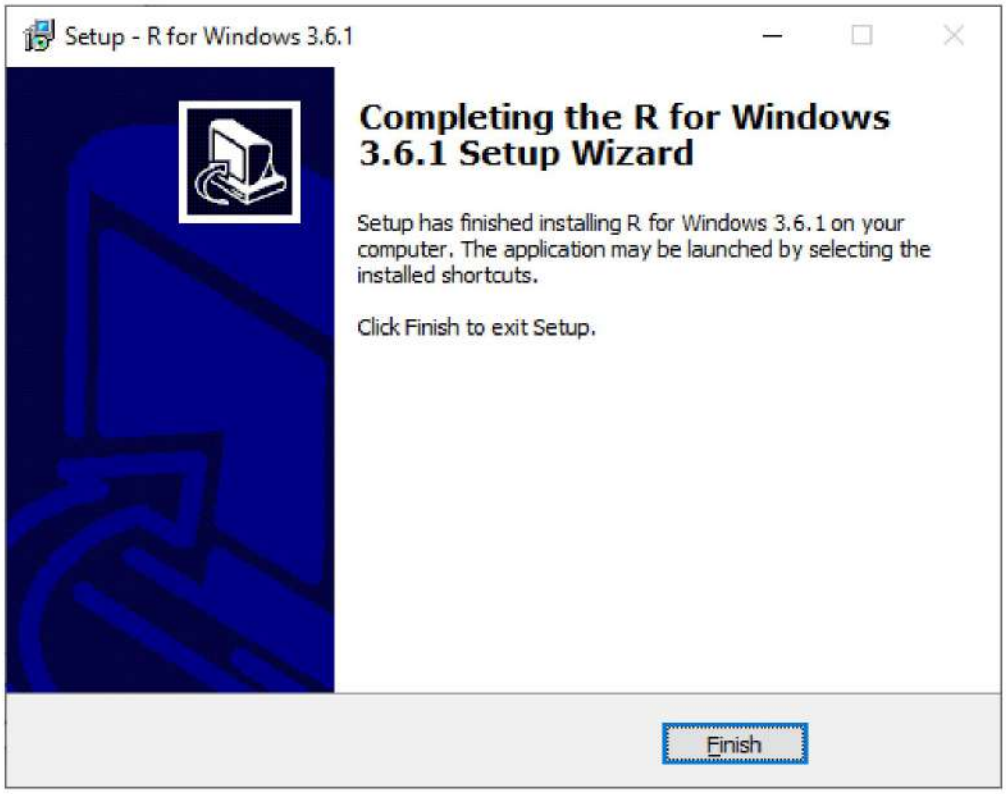
3) In the next step, we have to select either customized startup or accept the default, and then we proceed to Next.



4) When we proceed to next, our installation of R in our system will get started:



5) Atlast, we will click on finish to successfully install R in our system.



Now, We have successfully installed R 4.3.2 on our system.

Vectors

A vector is simply a list of items that are of the same type.
To combine the list of items to a vector, use the c() function and separate the items by a comma.

Example:

```
> player<-c("Joginder","Mahesh","Bhupendra","Mamta","Elvish_Bhai")
> player
[1] "Joginder" "Mahesh" "Bhupendra" "Mamta" "Elvish_Bhai"
```

Sequence

In R, the seq() function is used to generate a regular sequence of numbers.
seq(from, to, by)
The seq() function returns a regular sequence.

Parameter value
The seq() function takes three parameter values:
from: This represents where the sequence should begin.
to: This represents where the sequence should end.
by: This represents the step size or interval of the sequence.

Example:

```
> seq(from=2,to=18,by=3)
[1]  2  5  8 11 14 17
```

Repetition

In the R programming language, A very useful function for creating a vector by repeating a given number vector with the specified number of times is the rep().
The general structure of rep() :
rep(v1,n1).
Here, v1 is repeated n1 times.
R - replicate elements of vector

The forms of rep() functions :
rep(v1, times=)
rep(v1, each=)
rep(v1, length=)

Example :

```
> x<-c(1:4)
> rep(x,times=2)
[1] 1 2 3 4 1 2 3 4
> rep(x,each=2)
[1] 1 1 2 2 3 3 4 4
> rep(x,len=9)
[1] 1 2 3 4 1 2 3 4 1
```

EXERCISES

Exeersise 2.1 The Weights of five people before and after a diet programme are given in a table

Before	78	72	78	79	105
After	67	65	79	70	93

Read the before and after values into two different vector called Before and After . Use R to Evaluate the Amount of weight lost for each participant.
What is the Average amount of weight lost?

Solution :

```
> before<-c(78,72,78,79,105)
> after<-c(67,65,79,70,93)
> wt_lost<-before-after
> wt_lost
[1] 11  7 -1  9 12
```

Exercise 2.2 How would you write a function equivalent to sum((x-mean(x))^2) in a language like C or Java?

Solution :

```
#include <stdio.h>
double calculate_mean(double x[], int n) {
      double sum = 0.0;
      int i;
      for (i = 0; i < n; i++) {
         sum += x[i];
       } return sum / n;
}
double calculate_squared_difference(double x[], int n, double mean) {
      double sum_squared_diff = 0.0;
      int i;
      for (i = 0; i < n; i++) {
           sum_squared_diff += (x[i] - mean) * (x[i] - mean);
      } return sum_squared_diff;
}
int main() {
double x[] = {1.0, 2.0, 3.0, 4.0, 5.0};          // Sample array
int n = sizeof(x) / sizeof(x[0]);            // Number of elements in
array
double mean = calculate_mean(x, n);
double sum_squared_diff = calculate_squared_difference(x, n, mean);
printf("Sum of squared differences: %lf\n", sum_squared_diff);
return 0;
}
```

Exercise 2.3. Create the following vectors in R using `seq()` and `rep()`.

(i) $1, 1.5, 2, 2.5, \ldots, 12$

(ii) $1, 8, 27, 64, \ldots, 1000$.

(iii) $1, -\frac{1}{2}, \frac{1}{3}, -\frac{1}{4}, \ldots, -\frac{1}{100}$.

(iv) $1, 0, 3, 0, 5, 0, 7, \ldots, 0, 49$.

(v) $1, 3, 6, 10, 15, \ldots, \sum_{i=1}^{n} i, \ldots, 210$ [look up `?cumsum`].

(vi) * $1, 2, 2, 3, 3, 3, 4, \ldots, 9, \underbrace{10, \ldots, 10}_{10 \text{ times}}$. [Hint: type `?seq`, and read about the `times` argument.]

Solution :

```
> c(2:24)/2
 [1]  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5
8.0
[16]  8.5  9.0  9.5 10.0 10.5 11.0 11.5 12.0
> c(1:10)^3
 [1]    1    8   27   64  125  216  343  512  729 1000
> library(MASS)
> fractions(c(1,-1)/c(1:100))
  [1]      1   -1/2    1/3   -1/4    1/5   -1/6    1/7   -1/8    1/9  -1/10
 [11]   1/11  -1/12   1/13  -1/14   1/15  -1/16   1/17  -1/18   1/19  -1/20
 [21]   1/21  -1/22   1/23  -1/24   1/25  -1/26   1/27  -1/28   1/29  -1/30
 [31]   1/31  -1/32   1/33  -1/34   1/35  -1/36   1/37  -1/38   1/39  -1/40
 [41]   1/41  -1/42   1/43  -1/44   1/45  -1/46   1/47  -1/48   1/49  -1/50
 [51]   1/51  -1/52   1/53  -1/54   1/55  -1/56   1/57  -1/58   1/59  -1/60
 [61]   1/61  -1/62   1/63  -1/64   1/65  -1/66   1/67  -1/68   1/69  -1/70
 [71]   1/71  -1/72   1/73  -1/74   1/75  -1/76   1/77  -1/78   1/79  -1/80
 [81]   1/81  -1/82   1/83  -1/84   1/85  -1/86   1/87  -1/88   1/89  -1/90
 [91]   1/91  -1/92   1/93  -1/94   1/95  -1/96   1/97  -1/98   1/99 -1/100
> c(1:49)*c(1,0)
 [1]  1  0  3  0  5  0  7  0  9  0 11  0 13  0 15  0 17  0 19  0 21  0 23
0 25
[26]  0 27  0 29  0 31  0 33  0 35  0 37  0 39  0 41  0 43  0 45  0 47  0
49
Warning message:
In c(1:49) * c(1, 0) :
  longer object length is not a multiple of shorter object length
> cumsum(1:20)
 [1]   1   3   6  10  15  21  28  36  45  55  66  78  91 105 120 136 153
171 190
[20] 210
> rep(1:10,times=(1:10))
 [1]  1  2  2  3  3  3  4  4  4  4  5  5  5  5  5  6  6  6  6  6  6  7  7  7  7
[26]  7  7  7  8  8  8  8  8  8  8  8  9  9  9  9  9  9  9  9  9 10 10 10 10 10
[51] 10 10 10 10 10
```

Exercise 2.4. The $i^{\text{th term in Taylor Expansion of log(1+}}x)$ is $(-1)^{i+1}x^i/i$. Create a vector containing the first 100 terms for $x = 0.5$. *[Write out the first few entries by hand if that helps.]*

Let

$$r_n(x) = \log(1+x) - \sum_{i=1}^{n} \frac{(-1)^{i+1}x^i}{i}.$$

Evaluate $r_n(1)$ for $n = 10, 100, 1000, \ldots, 10^6$.

Solution :

```
> i<-c(1:100)
> sum((0.5^i)/i*c(1,-1))
[1] 0.4054651
> sum(c(1,-1)/c(1:10))
[1] 0.6456349
> sum(c(1,-1)/c(1:100))
[1] 0.6881722
> sum(c(1,-1)/c(1:1000))
[1] 0.6926474
> sum(c(1,-1)/c(1:10000))
[1] 0.6930972
> sum(c(1,-1)/c(1:100000))
[1] 0.6931422
> sum(c(1,-1)/c(1:1000000))
[1] 0.6931467
```

Conclusion-

Studied History and Installed a R 4.3.2 Programing Language. Also, Studied Vectors, Sequence and Series in R Successfully.

EXPERIMENT-2

Aim - To Study and Implementing the Subsetting and Logical Operation in R programming language.

Theory-

Subsetting

Subsetting is a way to extract specific elements from vectors, matrices, data frames, or lists based on certain conditions. In R , Use Square Bracket to select an individual or group of elements.

Example :

```
> numbers <- c(1, 2, 3, 4, 5)
> numbers[3]
[1] 3
> numbers[numbers > 2 & numbers < 5]
[1] 3 4
```

Logical Operators

In R, logical operators are used to perform logical operations on vectors, matrices, or data frames. These operators help evaluate conditions and produce logical (TRUE/FALSE) results.

Example :

```
> numbers>=2
[1] FALSE  TRUE  TRUE  TRUE  TRUE
> numbers!=3
[1]  TRUE  TRUE FALSE  TRUE  TRUE
```

Character Vector

In R, a character vector is a data structure that stores sequences of characters (i.e., strings). It is one of the basic data types in R and is commonly used for representing textual data.

Example :

```
> x<-c("Thara",1:3,"Bhai","Joginder")
[1] "Thara"    "1"        "2"        "3"        "Bhai"     "Joginder"
> x[5]
[1] "Bhai"
```

EXERCISES

Exercise 2.5 The built-in vector LETTERS contains the uppercase letters of the alphabet. Produce a vector of (i) the first 12 letters; (ii) the odd 'numbered' letters; (iii) the (English) consonants.

Solution :

```
> LETTERS
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q"
"R" "S"
[20] "T" "U" "V" "W" "X" "Y" "Z"
> LETTERS[1:12]
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L"
> LETTERS[seq(1,26,2)]
 [1] "A" "C" "E" "G" "I" "K" "M" "O" "Q" "S" "U" "W" "Y"
> vowels <- c("A", "E", "I", "O", "U")
>  LETTERS[!LETTERS %in% vowels]
 [1] "B" "C" "D" "F" "G" "H" "J" "K" "L" "M" "N" "P" "Q" "R" "S" "T" "V" "W" "X"
[20] "Y" "Z"
```

Exercise 2.6 The function **rnorm()** generates normal random variables. For instance, rnorm(10) gives a vector of 10 i.e standard normals. Generate 20 standard normals, and store them as x. Then obtain subvectors of
(i) the entries in x which are less than 1;
(ii) the entries between -1/2 and 1;
(iii) the entries whose absolute value is larger than 1.5.

Solution :

```
> x<-rnorm(20)
> x[x<1]
 [1]  0.08263036  0.98745716  0.38635087  0.70582575 -2.50258191 -0.12763076
 [7] -0.48041359  0.16005317 -0.25961240 -1.04547834  0.25860038  0.09265643
[13]  0.04075724 -1.57218130 -1.27010697  0.88059514  0.51503566 -0.62140655
> x[x>-1/2 & x<1]
 [1]  0.08263036  0.98745716  0.38635087  0.70582575 -0.12763076 -0.48041359
 [7]  0.16005317 -0.25961240  0.25860038  0.09265643  0.04075724  0.88059514
[13]  0.51503566
> x[abs(x)>1.5]
[1] -2.502582  2.010433  2.291611 -1.572181
```

Conclusion -

Studied and Implemented the Sub-Setting and Logical Operation in R programming language Successfully.

EXPERIMENT-3

Aim - To Study and Implementing the Matrices and Lists in R programming language.

Theory-

Matrices

In R, a matrix is a two-dimensional data structure that contains elements arranged in rows and columns. Matrices are useful for storing and manipulating data that is organized in a tabular format, such as datasets or mathematical matrices.

Example :

```
> matrix(c(1, 2, 3, 4, 5, 6), nrow = 3, ncol = 2)
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

Lists

In R, a list is a versatile data structure that can contain elements of different data types, such as numeric, character, logical, or even other lists. Lists are particularly useful for organizing and storing heterogeneous data or complex data structures.

Example :

```
> list(
+   numbers = c(1, 2, 3),
+   text = "Hello, world!",
+   logical = TRUE,
+   nested_list = list(a = 10, b = 20),
+   matrix = matrix(1:9, nrow = 3)
+ )
$numbers
[1] 1 2 3
$text
[1] "Hello, world!"
$logical
[1] TRUE
$nested_list
$nested_list$a
[1] 10
$nested_list$b
```

```
[1] 20
$matrix
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

EXERCISES

Exercise 2.7. Construct the matrix $B = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 2 & 6 \\ -3 & -1 & -3 \end{pmatrix}$

Show that $B \times B \times B$ is a scalar multiple of the identity matrix, and find the scalar.

Solution :

```
> B<-matrix(c(1,4,-3,2,2,-1,3,6,-3),3,3)
> B
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    2    6
[3,]   -3   -1   -3
> B%*%B%*%B
     [,1] [,2] [,3]
[1,]   -6    0    0
[2,]    0   -6    0
[3,]    0    0   -6
> diag(B%*%B%*%B)
[1] -6 -6 -6
```

Exercise 2.8. **Construct the following Matrices**

(a)

$$\begin{pmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{pmatrix}$$

(b)

$$\begin{pmatrix} 1 & -1 & 1 & \cdots & -1 \\ 1 & -1 & 1 & \cdots & -1 \\ \vdots & \vdots & & & \vdots \\ 1 & -1 & 1 & \cdots & -1 \end{pmatrix} \quad \text{(dimensions } 15 \times 10\text{)}.$$

(c) The **5X 15 matrix with three 1s in Shifting positions :**

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & & & & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 1 \end{pmatrix} \quad \text{(dimensions } 5 \times 15\text{)}.$$

d)

$$\begin{pmatrix} 1 & 2 & 3 & \cdots & 9 & 10 \\ 2 & 3 & 4 & \cdots & 10 & 11 \\ 3 & 4 & 5 & & & \vdots \\ \vdots & \vdots & & \ddots & & 17 \\ 9 & 10 & & & 17 & 18 \\ 10 & 11 & \cdots & 17 & 18 & 19 \end{pmatrix};$$

[Look at the `outer()` function.]

(e)

$$\begin{pmatrix} 1 & 2 & 3 & 4 & \cdots & 9 \\ 2 & 3 & 4 & & & 1 \\ 3 & 4 & \ddots & & & \vdots \\ 4 & & & \ddots & & 6 \\ \vdots & & & & 6 & 7 \\ 9 & 1 & \cdots & 6 & 7 & 8 \end{pmatrix};$$

%% may be useful here.]

(f)

$$\begin{pmatrix} I_5 & \mathbf{1} \\ \mathbf{0} & -I_6 \end{pmatrix}$$

where I_k is the $k \times k$-identity matrix, and $\mathbf{1}$ and $\mathbf{0}$ are matrices with all entries 1 and 0 respectively.

Solution :

```
> matrix(1:8,nrow=2)
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
> matrix(c(1,-1),15,10,byrow=TRUE)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1   -1    1   -1    1   -1    1   -1    1    -1
 [2,]    1   -1    1   -1    1   -1    1   -1    1    -1
 [3,]    1   -1    1   -1    1   -1    1   -1    1    -1
 [4,]    1   -1    1   -1    1   -1    1   -1    1    -1
 [5,]    1   -1    1   -1    1   -1    1   -1    1    -1
 [6,]    1   -1    1   -1    1   -1    1   -1    1    -1
 [7,]    1   -1    1   -1    1   -1    1   -1    1    -1
 [8,]    1   -1    1   -1    1   -1    1   -1    1    -1
 [9,]    1   -1    1   -1    1   -1    1   -1    1    -1
[10,]    1   -1    1   -1    1   -1    1   -1    1    -1
[11,]    1   -1    1   -1    1   -1    1   -1    1    -1
[12,]    1   -1    1   -1    1   -1    1   -1    1    -1
[13,]    1   -1    1   -1    1   -1    1   -1    1    -1
[14,]    1   -1    1   -1    1   -1    1   -1    1    -1
[15,]    1   -1    1   -1    1   -1    1   -1    1    -1
> r<-rep(c(1,0),times=c(3,15))
> matrix(r,5,15,byrow=TRUE)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15]
[1,]    1    1    1    0    0    0    0    0    0     0     0     0     0     0     0
[2,]    0    0    0    1    1    1    0    0    0     0     0     0     0     0     0
[3,]    0    0    0    0    0    0    1    1    1     0     0     0     0     0     0
[4,]    0    0    0    0    0    0    0    0    0     1     1     1     0     0     0
[5,]    0    0    0    0    0    0    0    0    0     0     0     0     1     1     1
> outer(1:10,0:9,"+")
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    2    3    4    5    6    7    8    9    10
 [2,]    2    3    4    5    6    7    8    9   10    11
 [3,]    3    4    5    6    7    8    9   10   11    12
 [4,]    4    5    6    7    8    9   10   11   12    13
 [5,]    5    6    7    8    9   10   11   12   13    14
 [6,]    6    7    8    9   10   11   12   13   14    15
 [7,]    7    8    9   10   11   12   13   14   15    16
 [8,]    8    9   10   11   12   13   14   15   16    17
 [9,]    9   10   11   12   13   14   15   16   17    18
[10,]   10   11   12   13   14   15   16   17   18    19
> e<-d [ d%%10!=0 & d%%11!=0 ]
> matrix(e%%10,9,9)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,]    1    2    3    4    5    6    7    8    9
 [2,]    2    3    4    5    6    7    8    9    1
 [3,]    3    4    5    6    7    8    9    1    2
 [4,]    4    5    6    7    8    9    1    2    3
 [5,]    5    6    7    8    9    1    2    3    4
 [6,]    6    7    8    9    1    2    3    4    5
 [7,]    7    8    9    1    2    3    4    5    6
 [8,]    8    9    1    2    3    4    5    6    7
 [9,]    9    1    2    3    4    5    6    7    8
> a<-diag(5)
> b<-matrix(1,5,6)
> c<-matrix(1,6,5)
> d<-diag(6)*(-1)
> e<-cbind(a,b)
> f<-cbind(c,d)
> rbind(e,f)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
 [1,]    1    0    0    0    0    1    1    1    1     1     1
 [2,]    0    1    0    0    0    1    1    1    1     1     1
 [3,]    0    0    1    0    0    1    1    1    1     1     1
 [4,]    0    0    0    1    0    1    1    1    1     1     1
 [5,]    0    0    0    0    1    1    1    1    1     1     1
 [6,]    1    1    1    1    1   -1    0    0    0     0     0
```

```
 [7,]    1    1    1    1    1    0   -1    0    0    0    0
 [8,]    1    1    1    1    1    0    0   -1    0    0    0
 [9,]    1    1    1    1    1    0    0    0   -1    0    0
[10,]    1    1    1    1    1    0    0    0    0   -1    0
[11,]    1    1    1    1    1    0    0    0    0    0   -1
```

Exercise 2.9 Solve the following system of simultaneous equations using matrix methods

$$\begin{aligned} a + 2b + 3c + 4d + 5e &= -5 \\ 2a + 3b + 4c + 5d + c &= 2 \\ 3a + 4b + 5c + d + 2e &= 5 \\ 4a + 5b + c + 2d + 3e &= 10 \\ 5a + b + 2c + 3d + 4e &= 11 \end{aligned}$$

Solution :

```
> d<-(0:35)
> a<-d[ d%%6!=0 & d%%7!=0 ]
> A<-matrix(a%%6,5,5)
> B<-c(-5,2,5,10,11)
> solve(A,B)
[1]  3.5066667  0.1066667 -0.6933333 -0.2933333 -1.0933333
```

Exercise 2.10 In this section we've seen that the behaviour of the function diag() depends upon its inputs. Can you think of some examples where this might cause a problem?

Solution :

```
> mat <- matrix(1:9, nrow = 3)  #Doesn't Showing a Diagonal Matrix
> diagonal_elements <- diag(mat)
> diag(mat)
[1] 1 5 9
> diag(c("a", "b", "c"))
     [,1] [,2] [,3]
[1,]   NA    0    0
[2,]    0   NA    0
[3,]    0    0   NA
Warning message:
In diag(c("a", "b", "c")) : NAs introduced by coercion
> diag()
Error in diag() : argument "nrow" is missing, with no default
```

Conclusion -

Studied and Implemented the Matrices and Lists in R programming language Successfully.

Exercise 1.1. Let x <- c(1,2,3) and y <- c(6,5,4). Predict what will happen when the following pieces of code are run. Check your answer.

a. x * 2
b. x * y
c. x[1] * y [2]

Solution :

```
> x<-c(1,2,3)
> y<-c(6,5,4)
> x*2
[1] 2 4 6
> x*y
[1]  6 10 12
> x[1]*y[2]
[1] 5
```

Exercise 1.2 Let x <- c(1,2,3) and y <- c(6,5,4). What is the value of x after each of the following commands? (Assume that each part starts with the values of x and y given above.)

a. x + x
b. x <- x + x
c. y <- x + x
d. x <- x + 1

Solution :

```
> x<-c(1,2,3)
> y<-c(6,5,4)
> x+x
[1] 2 4 6
> x
[1] 1 2 3
> x<-x+x
> x
[1] 2 4 6
> x<-c(1,2,3)
> y<-x+x
> x
[1] 1 2 3
> x<-x+1
> x
[1] 2 3 4
```

Exercise 1.3 Determine the values of the vector vec after each of the following commands is run.

a. vec <- 1:10
b. vec <- 1:10 * 2
c. vec <- 1:10^2
d. vec <- 1:10 + 1
e. vec <- 1: (10 * 2)
f. vec <- rep (c(1,1,2), times = 2)
g. vec <- seq(from= 0, to = 10, length.out = 5)

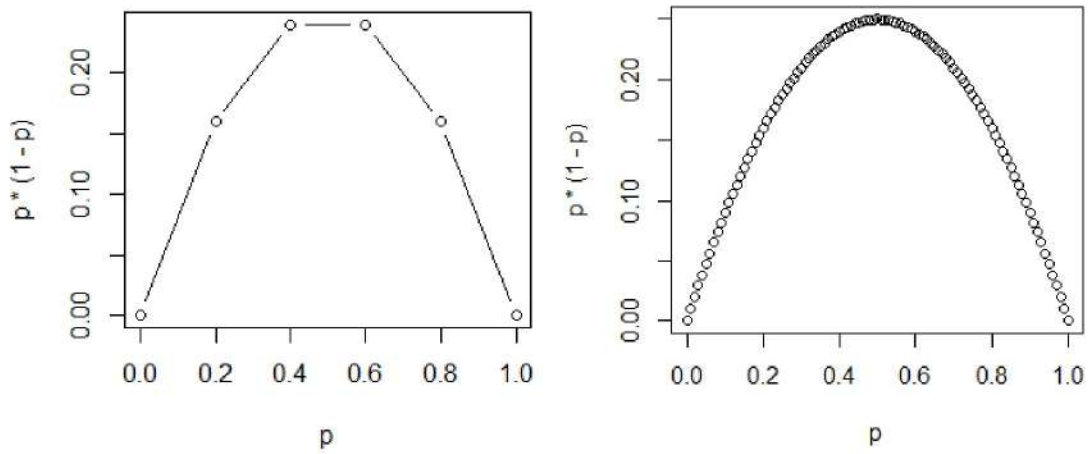
Solution :

```
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
> 1:10*2
 [1]  2  4  6  8 10 12 14 16 18 20
> 1:10^2
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
 [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
 [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
 [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
 [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
 [91]  91  92  93  94  95  96  97  98  99 100
> 1:10+1
 [1]  2  3  4  5  6  7  8  9 10 11
> 1:(10*2)
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
> rep(c(1,1,2),times=2)
[1] 1 1 2 1 1 2
> seq(from=0,to=10,length.out=5)
[1]  0.0  2.5  5.0  7.5 10.0
```

Exercise 1.4 In this exercise, you will graph the function f(p) = p(1-p) for p ∈ [0, 1].
a. Use seq to create a vector p of numbers from 0 to 1 spaced by 0.2.
b. Use plot to plot p in the x coordinate and p(1-p) in the y coordinate. Read the help page for plot and experiment with the type argument to find a good choice for this graph.
c. Repeat, but with creating a vector p of numbers from 0 to 1 spaced by 0.01.

Solution :

```
> p<-seq(0,1,0.2)
> plot(p,p*(1-p),type="b")
> p<-seq(0,1,0.01)
> plot(p,p*(1-p),type="b")
```



Exercise 1.5 Use R to calculate the sum of the squares of all numbers from 1 to 100: $1^2+2^2+...+ 99^2 + 100^2$.

Solution :

```
> sum((1:100)^2)
[1] 338350
```

Exercise 1.6 Let x be the vector obtained by running the R command x <- seq (from = 10, to = 30, by = 2).
a. What is the length of x?
b. What is x [2]?
c. What is x [1:5]?
d. What is x [1:3*2]?
e. What is x [1: (3*2)]?
f. What is x > 25?
g. What is x [x > 25]?
h. What is x[-1]?
i. What is x[-1:-3]?

Solution :

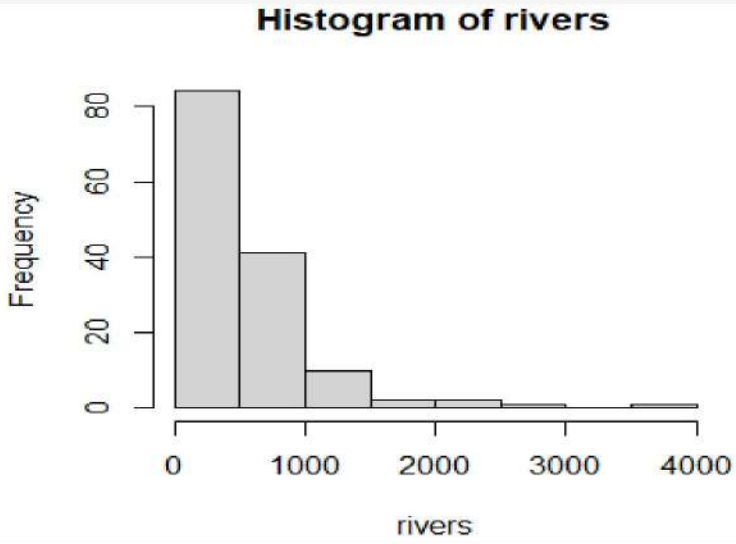
```
> x <- seq (from = 10, to = 30, by = 2)
> str(x)
 num [1:11] 10 12 14 16 18 20 22 24 26 28 ...
> length(x)
[1] 11
> x[2]
[1] 12
> x[1:5]
[1] 10 12 14 16 18
> x[1:3*2]
[1] 12 16 20
> x[1:(3*2)]
[1] 10 12 14 16 18 20
> x>25
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
> x[x>25]
[1] 26 28 30
> x[-1]
 [1] 12 14 16 18 20 22 24 26 28 30
> x[-1:-3]
[1] 16 18 20 22 24 26 28 30
```

Exercise 1.7 R has a built-in vector **rivers** which contains the lengths of major North American rivers.
a. Use ?rivers to learn about the data set.
b. Find the mean and standard deviation of the rivers data using the base R. functions mean and sd.
c. Make a histogram (hist) of the rivers data.
d. Get the five number summary (summary) of rivers data.
e. Find the longest and shortest lengths of rivers in the set.
f. Make a list of all (lengths of) rivers longer than 1000 miles.

Solution :

```
> ?rivers
```

```
> mean(rivers)
[1] 591.1844
> sd(rivers)
[1] 493.8708
> hist(rivers)
```




```
> summary(rivers)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  135.0   310.0   425.0   591.2   680.0  3710.0
> max(rivers)
[1] 3710
> min(rivers)
[1] 135
> rivers[rivers > 1000]
[1] 1459 1450 1243 2348 1171 3710 2315 2533 1306 1054 1270 1885 1100 1205 1038
[16] 1770
```

EXPERIMENT-4

Aim - To Study and Implementing the Data types, Missing Data, Data Frames in R programming language.

Theory-

Data Types

In R, there are several data types, each designed to store different types of information. Here are six fundamental data types in R:

Numeric: This data type is used to store numeric values, such as integers or floating-point numbers. Numeric values are represented by the numeric class in R.

Character: Character data type is used to store text or string values. Character values are enclosed in either single ('') or double ("") quotes.

Logical: The logical data type represents boolean values, which can be either TRUE or FALSE. Logical values are often used for comparisons and logical operations.

Integer: Integer data type specifically stores whole numbers. In R, integers are represented by the integer class. You can create integers using the L suffix.

Complex: Complex numbers consist of a real part and an imaginary part. They are represented by the complex class in R.

Raw: Raw data type is used to store binary data in its raw form. It is represented by the raw class.

Example :

```
> x<-3 #Numeric
> n<-"BhupendraJogi" #Character
> k<-TRUE      #logical
> i<-10L       #Integer
> z<-3+9i      #Complex
> r <- as.raw(c(0x48, 0x65, 0x6C, 0x6C, 0x6F))  #'Hello'in hex(Raw)
```

Missing Data

In R, missing data is represented by the special value NA, which stands for "Not Available". NA is used to denote missing values in vectors, matrices, data frames, and other data structures.

Example :

```
> x<-c(1,2,NA,4,NA)
> is.na(x)
[1] FALSE FALSE  TRUE FALSE  TRUE
> x[!is.na(x)]
[1] 1 2 4
> sum(x)
[1] NA
> x[is.na(x)]<-0
> sum(x)
[1] 7
```

Data Frame

In R, a data frame is a two-dimensional data structure that is similar to a table or spreadsheet. It is one of the most commonly used data structures in R and is particularly useful for storing and manipulating structured data, such as datasets.

Example :

```
> df <- data.frame(
+   Name = c("Alice", "Bob", "Charlie"),
+   Age = c(25, 30, 35),
+   Gender = c("Female", "Male", "Male")
+ )
> head(df)
     Name Age Gender
1   Alice  25 Female
2     Bob  30   Male
3 Charlie  35   Male
> tail(df)
     Name Age Gender
1   Alice  25 Female
2     Bob  30   Male
3 Charlie  35   Male
> print(df)
     Name Age Gender
1   Alice  25 Female
2     Bob  30   Male
3 Charlie  35   Male
> df[1,2]
[1] 25
> df$Name
[1] "Alice"   "Bob"     "Charlie"
> df[df$Age>30,]
     Name Age Gender
3 Charlie  35   Male
> merge(df,df,by="Name")
     Name Age.x Gender.x Age.y Gender.y
1   Alice    25   Female    25   Female
2     Bob    30     Male    30     Male
3 Charlie    35     Male    35     Male
> transform(df,NewColumn=Age*2)
     Name Age Gender NewColumn
1   Alice  25 Female        50
2     Bob  30   Male        60
3 Charlie  35   Male        70
> summary(df)
```

```
     Name                Age           Gender
 Length:3           Min.   :25.0   Length:3
 Class :character   1st Qu.:27.5   Class :character
 Mode  :character   Median :30.0   Mode  :character
                    Mean   :30.0
                    3rd Qu.:32.5
                    Max.   :35.0
```

Reading Data From Files

In R, you can read data from various types of files, such as CSV, Excel, text, and more, using specific functions designed for each file format.

Example :

```
> getwd()
[1] "C:/Users/Microsoft/Documents"
> setwd("C:/Users/Microsoft/Documents")
> data <- read.csv("file.csv")  # Reading CSV File
> library(readxl)
> data <- read_excel("file.xlsx") # Reading Excel File
```

EXERCISES

Exercise 1.8 Consider the built-in data frame **airquality**.

a. How many observations of how many variables are there?

b. What are the names of the variables?

c. What type of data is each variable?

d. Do you agree with the data type that has been given to each variable? What would have been some alternative choices?

Solution :

```
> str(airquality)
'data.frame':   153 obs. of  6 variables:
 $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
 $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
 $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day    : int  1 2 3 4 5 6 7 8 9 10 …
# Agreed with the data type given to each variable.
# Alternatively numeric and integer can be chosen interchangeably.
```

Exercise 1.9 There is a built-in data set **state**, which is really seven separate variables with names such as state.name, state.region, and state.area.

a. What are the possible regions a state can be in? How many states are in each region?

b. Which states have area less than 10,000 square miles?

c. Which state's geographic center is furthest south? (Hint: use which.min)

Solution :

```
> data(state)
```

```
> unique(state.region)
[1] South         West          Northeast     North Central
Levels: Northeast South North Central West
> table(state.region)
state.region
    Northeast         South North Central          West
            9            16            12            13
> state.name[state.area < 10000]
[1] "Connecticut"   "Delaware"      "Hawaii"        "Massachusetts"
[5] "New Hampshire" "New Jersey"    "Rhode Island"  "Vermont"
> state.name[which.min(state.center$y)]
[1] "Florida"
```

Exercise 1.10 Consider the **mtcars** data set.
a. Which cars have 4 forward gears?
b. What subset of mtcars does **mtcars [mtcars$disp > 150 & mtcars$mpg > 20,]** describe?
c. Which cars have 4 forward gears and manual transmission? (Note: manual transmission is 1 and automatic is 0.)
d. Which cars have 4 forward gears or manual transmission?
e. Find the mean mpg of the cars with 2 carburetors.

Solution :

```
> mtcars[mtcars$gear==4,]
                mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280       19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C      17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
> #mtcars [mtcars$disp > 150 & mtcars$mpg > 20,]
> #Describes the Cars Having more than 150 Displacement (cu.in.) and
more than 20 miles per gallon
> mtcars [mtcars$gear==4 & mtcars$am==1,]
                mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
```

```
Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
> mtcars [mtcars$gear==4 | mtcars$am==1,]
                mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280       19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C      17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora  15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
> m<-mtcars[mtcars$carb==2,]
> mean(m$mpg)
[1] 22.4
```

Exercise 1.11 Consider the **mtcars** data set.
a. Convert the am variable to a factor with two levels, auto and manual, by typing the following: mtcars$am <- factor(mtcars$am, levels = c(0, 1), labels = c("auto", "manual")).
b. How many cars of each type of transmission are there?
c. How many cars of each type of transmission have gas mileage estimates greater than 25 mpg?

Solution :

```
> mtcars$am <-factor(mtcars$am,levels=c(0,1),labels=c("auto","manual"))
> table(mtcars$am)
 0  1
19 13
> table(mtcars$am[mtcars$mpg>25])
1
6
```

Conclusion -

Studied and Implemented the Data types, Missing Data, Data Frames in R programming language Successfully.

EXPERIMENT-5

Aim - To Study and Implementing the Packages, Errors and Warnings in R programming language.

Theory-

Packages

In R, packages are collections of functions, data, and documentation that extend the capabilities of base R. They provide additional functionality for specific tasks such as data manipulation, statistical analysis, visualization, and more. You can install packages from CRAN (Comprehensive R Archive Network) using the **install.packages()** function and then load them into your R session using the **library()** function.

ggplot2: For creating elegant and complex visualizations.
dplyr: For data manipulation tasks such as filtering, selecting, and summarizing data.
tidyr: For reshaping and tidying messy data.
caret: For machine learning tasks such as building predictive models.
magrittr: For creating readable and expressive code using the pipe operator %>%.
readr: For efficient reading of structured text data.

Example :

```
> install.packages("ggplot2")
also installing the dependencies 'colorspace', 'farver', 'labeling',
'munsell', 'RColorBrewer', 'viridisLite', 'gtable', 'isoband',
'scales', 'withr'
trying URL
'https://cran.icts.res.in/bin/windows/contrib/4.3/colorspace_2.1-0.zip'
Content type 'application/zip' length 2632991 bytes (2.5 MB)
downloaded 2.5 MB
trying URL
'https://cran.icts.res.in/bin/windows/contrib/4.3/farver_2.1.1.zip'
Content type 'application/zip' length 1505817 bytes (1.4 MB)
downloaded 1.4 MB
trying URL
'https://cran.icts.res.in/bin/windows/contrib/4.3/labeling_0.4.3.zip'
Content type 'application/zip' length 62568 bytes (61 KB)
downloaded 61 KB
trying URL
'https://cran.icts.res.in/bin/windows/contrib/4.3/munsell_0.5.0.zip'
Content type 'application/zip' length 244738 bytes (239 KB)
downloaded 239 KB
```

```
trying URL
'https://cran.icts.res.in/bin/windows/contrib/4.3/RColorBrewer_1.1-3.zi
p'
Content type 'application/zip' length 56066 bytes (54 KB)
downloaded 54 KB
trying URL
'https://cran.icts.res.in/bin/windows/contrib/4.3/viridisLite_0.4.2.zip
'
Content type 'application/zip' length 1300092 bytes (1.2 MB)
downloaded 1.2 MB
trying URL
'https://cran.icts.res.in/bin/windows/contrib/4.3/gtable_0.3.4.zip'
Content type 'application/zip' length 225727 bytes (220 KB)
downloaded 220 KB
trying URL
'https://cran.icts.res.in/bin/windows/contrib/4.3/isoband_0.2.7.zip'
Content type 'application/zip' length 1968436 bytes (1.9 MB)
downloaded 1.9 MB
trying URL
'https://cran.icts.res.in/bin/windows/contrib/4.3/scales_1.3.0.zip'
Content type 'application/zip' length 703883 bytes (687 KB)
downloaded 687 KB
trying URL
'https://cran.icts.res.in/bin/windows/contrib/4.3/withr_3.0.0.zip'
Content type 'application/zip' length 245992 bytes (240 KB)
downloaded 240 KB
trying URL
'https://cran.icts.res.in/bin/windows/contrib/4.3/ggplot2_3.5.0.zip'
Content type 'application/zip' length 4822730 bytes (4.6 MB)
downloaded 4.6 MB
package 'colorspace' successfully unpacked and MD5 sums checked
package 'farver' successfully unpacked and MD5 sums checked
package 'labeling' successfully unpacked and MD5 sums checked
package 'munsell' successfully unpacked and MD5 sums checked
package 'RColorBrewer' successfully unpacked and MD5 sums checked
package 'viridisLite' successfully unpacked and MD5 sums checked
package 'gtable' successfully unpacked and MD5 sums checked
package 'isoband' successfully unpacked and MD5 sums checked
package 'scales' successfully unpacked and MD5 sums checked
package 'withr' successfully unpacked and MD5 sums checked
package 'ggplot2' successfully unpacked and MD5 sums checked
The downloaded binary packages are in
```

```
C:\Users\Microsoft\AppData\Local\Temp\RtmpAXs3qw\downloaded_packages
> library(ggplot2)
```

Errors and Warnings

In R, errors and warnings are messages that indicate issues encountered during the execution of code. **Errors** occur when R encounters a problem that prevents it from executing the code, while **warnings** indicate potential issues that do not halt the execution but may affect the results. It's important to understand and address errors and warnings in your code to ensure correctness and reliability. You can use functions like **tryCatch()** to handle errors gracefully and **suppressWarnings()** to ignore specific warnings and **suppressMessages()** to ignore specific Messages.

Example :

```
> airquality[airquality$Month=4,]
Error: unexpected '=' in "airquality[airquality$Month="
> v<-1:10
> d<-1:3
> v*d
 [1]  1  4  9  4 10 18  7 16 27 10
Warning message:
In v * d : longer object length is not a multiple of shorter object
length
> suppressWarnings(v*d)
 [1]  1  4  9  4 10 18  7 16 27 10
```

Useful Idioms

There are several idioms and best practices in R that can help you write more efficient, readable, and concise code. Some Useful Idioms are:
Vectorization,Function Composition,Tidy Data Principles,Avoiding Global Variables,Documentation and Comments

Example :

```
> v<-c(2,3,4,-4,8,7,5,2,3,-1,5,9,8,1,2,5,6,-7,-6)
> sum(v==3)
[1] 2
> mean(v==3)
[1] 0.1052632
> table(v)
v
-7 -6 -4 -1  1  2  3  4  5  6  7  8  9
 1  1  1  1  1  3  2  1  3  1  1  2  1
> max(table(v))
[1] 3
> length(unique(v))
[1] 13
> v[v>0]
```

```
 [1] 2 3 4 8 7 5 2 3 5 9 8 1 2 5 6
> v<-c(2,3,4,NA,-4,8,7,5,2,3,-1,NA,5,9,8,1,2,5,6,-7,-6)
> v[!is.na(v)]
 [1]  2  3  4 -4  8  7  5  2  3 -1  5  9  8  1  2  5  6 -7 -6
```

EXERCISES

Exercise 1.12 This problem uses the data set **hot_dogs** from the package **fosdata.**

a. How many observations of how many variables are there? What types are the variables?

b. What are the three kinds of hot dogs in this data set?

c. What is the highest sodium content of any hot dog in this data set?

d. What is the mean calorie content for Beef hot dogs?

Solution :

```
> library(fosdata)
> nrow(hot_dogs)
[1] 54
> ncol(hot_dogs)
[1] 3
> sapply(hot_dogs,class)
     type  calories    sodium
 "factor" "integer" "integer"
> unique(hot_dogs$type)
[1] Beef    Meat    Poultry
Levels: Beef Meat Poultry
> max(hot_dogs$sodium)
[1] 645
> mean(hot_dogs[hot_dogs$type=="Beef",]$calories)
[1] 156.85
```

Exercise 1.13 This problem uses the data set **DrinksWages** from the package **HistData.**

a. How many observations of how many variables are there? What types are the variables?

b. The variable wage contains the average wage for each profession. Which profession has the lowest wage?

c. The variable n contains the number of workers surveyed for each profession. Sum this to find the total number of workers surveyed.

d. Compute the mean wage for all workers surveyed by multiplying wage * n for each profession, summing, and dividing by the total number of workers surveyed.

Solution :

```
> library(HistData)
> str(DrinksWages)
'data.frame':   70 obs. of  6 variables:
 $ class : Factor w/ 3 levels "A","B","C": 1 1 1 1 1 1 1 1 1 1 ...
 $ trade : Factor w/ 70 levels "baker","barman",..: 38 10 25 55 36 44 68 34 14
11 ...
 $ sober : int  1 1 2 1 2 9 8 3 0 12 ...
```

```
 $ drinks: int  1 10 1 5 0 8 2 5 7 23 ...
 $ wage  : num  24 18.4 21.5 21.2 19 ...
 $ n     : int  2 11 3 6 2 17 10 8 7 35 ...
> DrinksWages[DrinksWages$wage==min(DrinksWages$wage),"trade"]
[1] factory worker
> sum(DrinksWages$n)
[1] 604
> sum(DrinksWages$wage*DrinksWages$n)/sum(DrinksWages$n)
[1] 24.59782
```

Exercise 1.14 This problem uses the package **Lahman**. The data set **Batting**, in the Lahman package contains batting statistics of all major league baseball players since 1871, broken down by season.
a. How many observations of how many variables are there?
b. Use the command head (Batting) to get a look at the first six lines of data.
c. What is the most number of triples (X3B) that have been hit in a single season?
d. What is the playerID(s) of the person(s) who hit the most number of triples in a single season? In what year did it happen?
e. Which player hit the most number of triples in a single season since 1960?

Solution :

```
> library(Lahman)
> nrow(Batting)
[1] 112184
> ncol(Batting)
[1] 22
> head(Batting)
   playerID yearID stint teamID lgID  G  AB  R  H X2B X3B HR RBI SB CS BB SO IBB  HBP SH SF GIDP
1 abercda01   1871     1    TRO   NA  1   4  0  0   0   0  0   0  0  0  0  0  NA   NA NA NA    0
2  addybo01   1871     1    RC1   NA 25 118 30 32   6   0  0  13  8  1  4  0  NA   NA NA NA    0
3 allisar01   1871     1    CL1   NA 29 137 28 40   4   5  0  19  3  1  2  5  NA   NA NA NA    1
4 allisdo01   1871     1    WS3   NA 27 133 28 44  10   2  2  27  1  1  0  2  NA   NA NA NA    0
5 ansonca01   1871     1    RC1   NA 25 120 29 39  11   3  0  16  6  2  2  1  NA   NA NA NA    0
6 armstbo01   1871     1    FW1   NA 12  49  9 11   2   1  0   5  0  1  0  1  NA   NA NA NA    0
> library(dplyr)
> library(magrittr)
> s<-Batting %>% group_by(yearID) %>% summarise(NoOfTriples =
sum(X3B),)
> max(s$NoOfTriples)
[1] 1895
> m<-max(Batting$X3B)
> Batting[Batting$X3B==m,c("X3B","yearID")]
      X3B yearID
13806  36   1912
> m<-max(Batting[Batting$yearID>=1960,]$X3B)
> B<-Batting[Batting$yearID>=1960,]
> m<-max(B$X3B)
```

```
> B[B$X3B==m,"playerID"]
[1] "grandcu01"
```

Exercise 1.15 Consider the **bechdel** data set in the **fosdata** package.
a. How many movies in the data set pass the Bechdel test?
b. What percentage of movies in the data set pass the Bechdel test?
c. Create a table of number of movies in the data set by year.
d. Which year has the most movies in the data set?
e. How many different values are there in the clean_test variable?
f. Create a data frame that contains only those observations that pass the Bechdel test.
g. Create a data frame that contains all of the observations that do not have missing values in the domgross variable.

Solution :

```
> library(fosdata)
> sum(bechdel$binary=="PASS")
[1] 803
> mean(bechdel$binary=="PASS")
[1] 0.4476031
> table(bechdel$year)
1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985
   1    5    3    5    7    5    8    7    8    5   14    9   14    5   16   10
1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001
  10   14   19   14   15   13   20   16   26   36   42   51   62   56   63   64
2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013
  80   64   81  100   90   73  101  124  129  124   86   99
> max(table(bechdel$year))
[1] 129
> length(unique(bechdel$clean_test))
[1] 5
> PassOnly<-bechdel[bechdel$binary=="PASS",]
> NonMissingOnly<-bechdel[!(is.na(bechdel$domgross)), ]
```

Conclusion -

Studied and Implemented the Packages, Errors and Warnings in R programming language Successfully.

EXPERIMENT-6

Aim - To Study and Implementing the Centrality, Spread, Count, %age, Proportions, 5 Number Summary in R programming language.

Theory-

Centrality: Represents the typical or central value in a dataset.

Mean : Average of all values in the dataset.
Median : Middle value when the dataset is ordered.
Mode : Most frequently occurring value(s) in the dataset.

Counts : Total number of data points in the dataset.
Percentages : Portions of the dataset expressed in relation to the total.
Proportions : Relative sizes of different categories within the dataset.

Quantile : Values that divide the dataset into four equal parts.
Percentile : Specific point in a dataset below which a given percentage of values falls.
5 Number Summary: Minimum, first quartile, median, third quartile, and maximum values in the dataset.

Spread : Measure of how dispersed/spread out the values in the dataset are.

Variance: Average of the squared differences from the mean.
Standard Deviation: Measure of the amt of variation/dispersion of a set of values.
Interquartile Range: Range between the first and third quartiles, representing the middle 50% of the dataset.

Example :

```
> data<-c(6,34,12,8.6,7,14.6,28.6,7,19,12,7,14.6,98)
> mean(data)
[1] 20.64615
> median(data)
[1] 12
> mode(data)
[1] "numeric"
> t<-table(data)
> t[t==max(t)]
```

```
7
3
> t
data
   6    7  8.6   12 14.6   19 28.6   34   98
   1    3    1    2    2    1    1    1    1
> table(data)/length(data)
data
        6         7       8.6        12      14.6        19      28.6        34        98
0.07692308 0.23076923 0.07692308 0.15384615 0.15384615 0.07692308 0.07692308 0.07692308 0.07692308
> round(table(data)/length(data)*100,digits=1)
data
   6    7  8.6   12 14.6   19 28.6   34   98
 7.7 23.1  7.7 15.4 15.4  7.7  7.7  7.7  7.7
> quantile(data,prob=c(0.5))
50%
 12
> summary(data)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   6.00    7.00   12.00   20.65   19.00   98.00
> var(data)
[1] 614.0677
> sd(data)
[1] 24.78039
> IQR(data)
[1] 12
```

☐ EXERCISES ☐

Exercise 13.2.a Obtain, rounded to two decimal places, the proportion of seismic events in the quakes data frame that occurred at a depth of 300 km or deeper.

Solution :

```
> round(sum(quakes$depth>=300)/nrow(quakes),digits=2)
[1] 0.45
```

Exercise 13.2.b Remaining with the quakes data set, calculate the mean and median magnitudes of the events that occurred at a depth of 300 km or deeper.

Solution :

```
> mean(quakes$mag[quakes$depth>=300])
[1] 4.527373
> median(quakes$mag[quakes$depth>=300])
[1] 4.5
```

Exercise 13.2.c Using the chickwts data set, write a for loop that gives you the mean weight of chicks for each feed type-the same as the results given by the tapply function in Section 13.2.1. Display the results rounded to one decimal place and, when printing, ensure each mean is labeled with the appropriate feed type.

Solution :

```
> means <- list()
> for (i in unique(chickwts$feed)) {
+     means[[i]] <- mean(chickwts$weight[chickwts$feed == i])
+ }
> print(unlist(means))
horsebean   linseed   soybean sunflower  meatmeal    casein
 160.2000  218.7500  246.4286  328.9167  276.9091  323.5833
```

-->InsectSprays contains data on the number of insects found on various agricultural units, as well as the type of insect spray that was used on each unit. Ensure you can access the data frame at the prompt; then study the help file ?lnsectSprays to get an idea of R's representation of the two variables.

Exercise 13.2.d Identify the two variable types in InsectSprays (as per the definitions in Section 13.1.1 and Section 13.1.2)

Solution :

```
> str(InsectSprays)                    // Categorical and Numerical
Variables
'data.frame':  72 obs. of  2 variables:
 $ count: num  10 7 20 14 14 12 10 23 17 20 ...
 $ spray: Factor w/ 6 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1
...
```

Exercise 13.2.e Calculate the modes of the distribution of insect counts, regardless of spray type.

Solution :

```
> mode_counts <- table(InsectSprays$count)
> names(mode_counts)[mode_counts == max(mode_counts)]
[1] "3"
```

Exercise 13.2.f Use tapply to report the total insect counts by each spray type.

Solution :

```
>  tapply(InsectSprays$count, InsectSprays$spray, sum)
  A   B   C   D   E   F
174 184  25  59  42 200
```

Exercise 13.2.g Using the same kind of for loop as in (c), compute the percentage of agricultural units in each spray type group that had at least five bugs on them. When printing to the screen, round the percentages to the nearest whole number.

Solution :

```
> percentages <- list()
> for (i in unique(InsectSprays$spray)) {
+     typei <- InsectSprays[InsectSprays$spray == i, ]
+     percentage <- round( sum(typei$count >= 5) / nrow(typei) * 100 )
+     percentages[[i]] <- percentage }
> print(percentages)
$A
[1] 100
$B
[1] 100
$C
[1] 8
$D
[1] 58
$E
[1] 33
$F
[1] 100
```

Exercise 13.2.h Obtain the same numeric results as in (g), with rounding, but use tapply and a disposable function.

Solution :

```
> percentages <- tapply(InsectSprays$count, InsectSprays$spray,
FUN=function(x) round( sum(x >= 5) / length(x) * 100 ))
> print(percentages)
  A   B   C   D   E   F
100 100   8  58  33 100
```

Exercise 13.3.a Using the chickwts data frame, compute the 10th, 30th, and 90th percentiles of all the chick weights and then use tapply to determine which feed type is associated with the highest sample variance of weights.

Solution :

```
> quantile(chickwts$weight,prob=c(0.1,0.3,0.9))
10% 30% 90%
153 217 359
>
> tapply(chickwts$weight , chickwts$feed , var)
   casein horsebean   linseed  meatmeal   soybean sunflower
 4151.720  1491.956  2728.568  4212.091  2929.956  2384.992
```

Exercise 13.3.b Turn to the seismic event data in quakes and complete the following tasks:
i. Find the IQR of the recorded depths.
ii. Find the five-number summary of all magnitudes of seismic events that occur at a depth of 400 km or deeper. Compare this to the summary values found in Section 13.2.3 of those events occurring at less than 400 km and briefly comment on what you notice.
iii. Use your knowledge of cut (Section 4.3.3) to create a new factor vector called **depthcat** that identifies four evenly spaced categories of quakes$depth **iv.**Find the sample mean and standard deviation of the magnitudes of the events associated with each category of depth according to depthcat.
v.Use tapply to compute the 0.8th quantile of the magnitudes of the seismic events in quakes, split by depthcat.

Solution :

```
 > IQR(quakes$depth)
[1] 444
> summary(quakes$mag[quakes$depth<400])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   4.00    4.40    4.60    4.67    4.90    6.40
> summary(quakes$mag[quakes$depth>=400])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.000   4.200   4.500   4.545   4.700   5.900
> // On increasing depth , We noticed that the mean mag decreases along
with the min max Q1 Q2 Q3

> depthcat <- cut(quakes$depth, breaks = c(40, 200, 360, 520, 680),
include.lowest = TRUE)
> levels(depthcat)
[1] "[40,200]"  "(200,360]" "(360,520]" "(520,680]"
> tapply(quakes$mag,depthcat,function(x) c(mean = mean(x), sd = sd(x)))
$`[40,200]`
     mean        sd
4.7358852 0.4038238
$`(200,360]`
     mean        sd
4.5484277 0.3783467
$`(360,520]`
     mean        sd
4.4992000 0.3617823
$`(520,680]`
     mean        sd
4.5476510 0.3909901
> tapply(quakes$mag, depthcat, function(x) quantile(x, probs = 0.8))
```

```
[40,200] (200,360] (360,520] (520,680]
     5.1       4.8       4.8       4.9
```

Conclusion -

Studied and Implemented the Centrality, Spread, Count, %age, Proportions, 5 Number Summary in R programming language Successfully.

EXPERIMENT-7

Aim - To Study and Implementing the Covariance Correlation and Outliers in R programming language.

Theory-

Covariance:
Covariance measures the degree to which two variables change together.
A positive covariance indicates that as one variable increases, the other variable tends to increase as well, and vice versa for negative covariance.
However, the magnitude of covariance depends on the scales of the variables, making it difficult to interpret directly.

Correlation:
Correlation is a standardized measure of the linear relationship between two variables.
It ranges from -1 to 1, where 1 indicates a perfect positive linear relationship, -1 indicates a perfect negative linear relationship, and 0 indicates no linear relationship.
Correlation is preferred over covariance because it is scale-independent and easier to interpret.

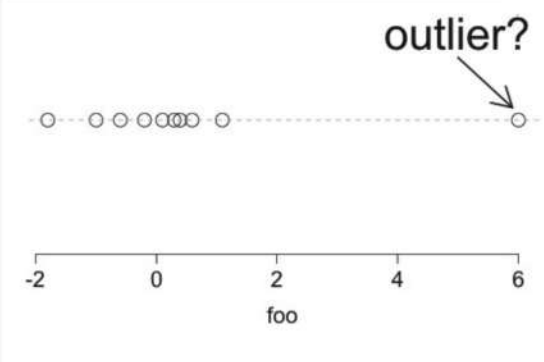
Outliers:
Outliers are data points that significantly deviate from the rest of the data in a dataset.
They can arise due to measurement errors, natural variation, or anomalies in the data.
It's essential to identify and understand outliers to decide whether to exclude them from the analysis or investigate them further to determine their cause.

Example :

```
> x<-c(2,4.4,3,3,2,2.2,2,4)
> y<-c(1,4.4,1,3,2,2.2,2,7)

> cov(x,y)
[1] 1.479286

> cov(x,y)/(sd(x)*sd(y))
[1] 0.7713962
> cor(x,y)
[1] 0.7713962
```




```
>plot(foo,rep(0,10),yaxt="n",ylab="",bty="n",cex=1,cex.axis=1.5,cex.lab
=1.5)
```

```
> abline(h=0,col="blue",lty=2)
> arrows(5,0.5,5.9,0.1,lwd=2)
> text(5,0.1,labels="outlier?",cex=3)
```

EXERCISES

Exercise 13.4.a In Exercise 7.1 (b) on page 139, you plotted height against weight measurements. Compute the correlation coefficient based on the observed data of these two variables.

Solution :

```
> height <- c(160, 165, 170, 175, 180)
> weight <- c(60, 65, 70, 75, 80)
> cor(height, weight)
[1] 1
```

Exercise 13.4.b Another of R's built-in, ready-to-use data sets is mtcars containing a number of descriptive details on performance aspects of 32 automobiles.

i. Ensure you can access this data frame by entering mtcars at the prompt. Then inspect its help file to get an idea of the types of data present.

ii. Two of the variables describe a vehicle's horsepower and shortest time taken to travel a quarter-mile distance. Using base R graphics, plot these two data vectors with horsepower on the x-axis and compute the correlation coefficient.

iii. Identify the variable in mtcars that corresponds to transmission type. Use your knowledge of factors in R to create a new factor from this variable called tranfac, where manual cars should be labelled "manual" and automatic cars "auto".

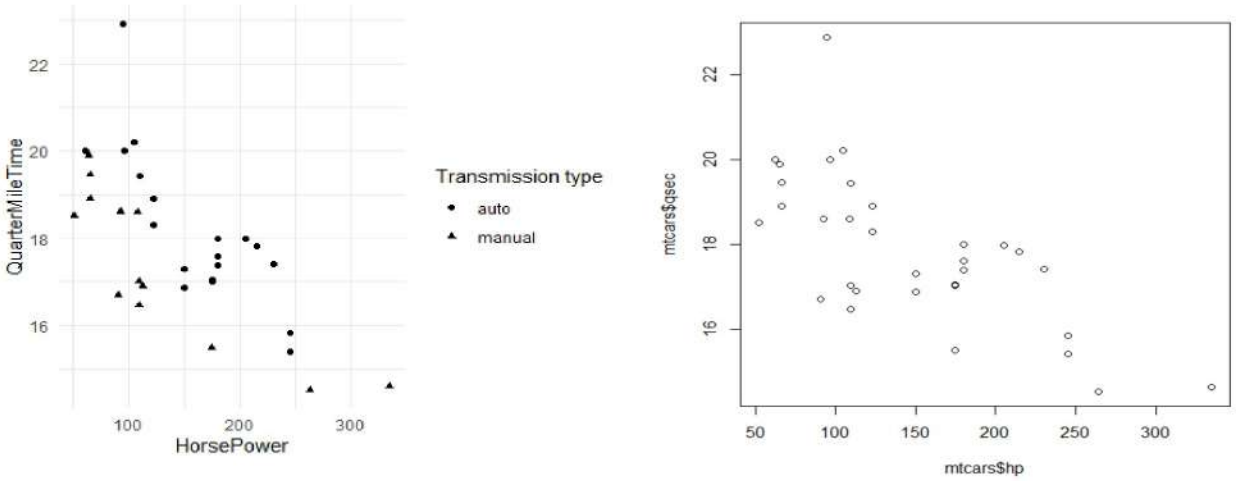
iv. Now, use ggplot from ggplot2 in conjunction with tranfac to produce the same scatterplot as in (ii) so that you're able to visually differentiate between manual and automatic cars.

v. Finally, compute separate correlation coefficients for horse power and quarter-mile time based on the transmission of the vehicles and, comparing these estimates with the overall value from (ii), briefly comment on what you note.

Solution :

```
> help(mtcars)
> ?mtcars
> plot(mtcars$hp,mtcars$qsec)
> cor(mtcars$hp,mtcars$qsec)
[1] -0.7082234
> mtcars$tranfac <- factor(mtcars$am, levels = c(0, 1), labels =
c("auto", "manual"))
> View(mtcars)
> library(ggplot2)
```

```
> ggplot(data = mtcars, aes(x = hp, y = qsec, shape = tranfac)) +
+     geom_point() +
+     labs(x = "HorsePower", y = "QuarterMileTime", shape =
"Transmission type") + theme_minimal()
> cor(mtcars[mtcars$am==0,]$hp,mtcars[mtcars$am==0,]$qsec)
[1] -0.8040275
> cor(mtcars[mtcars$am==1,]$hp,mtcars[mtcars$am==1,]$qsec)
[1] -0.8494566
> #These differences suggest that the relationship between horsepower
and quarter-mile time may vary depending on the transmission type.
Manual cars tend to have a stronger negative correlation between
horsepower and quarter-mile time compared to automatic cars
```



Exercise 13.4.c Return to chickwts to complete the following tasks:

i. Produce a plot like the left panel of Figure 13-7, based on the weights of chicks on the sunflower diet only. Note that one of the sunflower-fed chicks has a far lower weight than the others.

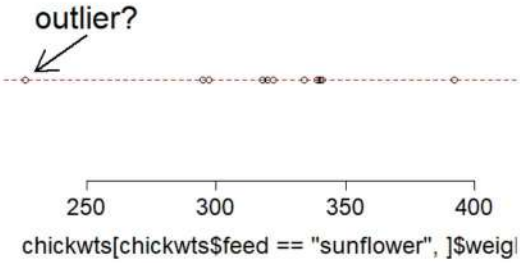
ii. Compute the standard deviation and IQR of the weights of the sunflower-fed chicks.

iii. Now, suppose you're told that the lowest weight of the sunflower-fed chicks was caused by a certain illness, irrelevant to your research. Delete this observation and recalculate the standard deviation and IQR of the remaining sunflower chicks. Briefly comment on the difference in calculated values.

Solution :

```
> plot( chickwts[chickwts$feed=="sunflower",]$weight , rep(0,12) ,
yaxt="n", ylab="", bty="n",cex=1,cex.axis=1.5, cex.lab=1.5)
> abline(h=0,col="red",lty=2)
> arrows(250, 0.5 , 230 , 0.1 ,lwd=2)
> text(250,0.7,labels="outlier?",cex=2)
> sd( chickwts[chickwts$feed=="sunflower",]$weight )
```

```
[1] 48.83638
> IQR( chickwts[chickwts$feed=="sunflower",]$weight )
[1] 27.5
> lowest <- which.min(chickwts[chickwts$feed == "sunflower", "weight"])
> A<-chickwts[chickwts$feed == "sunflower", "weight"]
> Achickwts <- A[-lowest]
```




```
> sd(Achickwts)
[1] 38.31473
> IQR(Achickwts)
[1] 21.5
> #Removing an outlier reduces the variability
and spread of the data, resulting in smaller
standard deviation and IQR values.
```

Conclusion -

Studied and Implemented the Covariance Correlation and Outliers in R programming language Successfully.

EXPERIMENT-8

Aim - To Study and Implementing the Various Data Visualisation Plots in R programming language.

Theory-

Barplot:

A barplot is used to display the frequency or distribution of categorical data by representing each category as a bar with the height proportional to its frequency.

Syntax:

```
barplot(height, names.arg = categories, xlab = "X-axis label", ylab = "Y-axis label", main = "Title")
```

height: Numeric vector containing the heights of the bars.
names.arg: Optional vector specifying the names of the categories.
xlab: Label for the x-axis.
ylab: Label for the y-axis.
main: Title for the plot.

Pie Chart:

A pie chart is used to display the distribution of categorical data as a circular graph divided into slices, with each slice representing a category and its size proportional to its frequency.

Syntax:

```
pie(x, labels = categories, main = "Title")
```

x: Numeric vector containing the frequencies of each category.
labels: Optional vector specifying the labels for each slice.
main: Title for the plot.

Histograms:

A histogram is used to represent the distribution of continuous data by dividing the data into bins and displaying the frequency of observations falling into each bin.

Syntax:

```
hist(x, breaks = "Sturges", xlab = "X-axis label", ylab = "Y-axis label", main = "Title")
```

x: Numeric vector containing the data.
breaks: Method to determine the no. of bins ("Sturges" is a common method). xlab: Label for the x-axis.
ylab: Label for the y-axis.
main: Title for the plot.

Boxplot:

A boxplot (box-and-whisker plot) is used to display the distribution of continuous data by summarizing key features such as the median, quartiles, and outliers.

Syntax:

```
boxplot(x, horizontal = FALSE, xlab = "X-axis label", ylab = "Y-axis label", main = "Title")
```

x: Numeric vector containing the data.
horizontal: Logical value indicating whether to plot a horizontal boxplot.
xlab: Label for the x-axis.
ylab: Label for the y-axis.
main: Title for the plot.

Scatterplot:

A scatterplot is used to display the relationship between two continuous variables by plotting each observation as a point on a graph, with one variable on the x-axis and the other on the y-axis.

Syntax:

```
plot(x, y, xlab = "X-axis label", ylab = "Y-axis label", main = "Title")
```

x: Numeric vector containing the values for the x-axis.
y: Numeric vector containing the values for the y-axis.
xlab: Label for the x-axis.
ylab: Label for the y-axis.
main: Title for the plot.

☐ EXERCISES

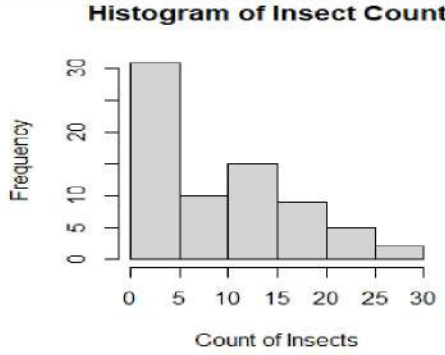
☐

Recall the built-in InsectSprays data frame, containing counts of insects on various agricultural units treated with one of six sprays.

Exercise 14.1.a Produce a histogram of the counts of insects using base R graphics.

Solution :

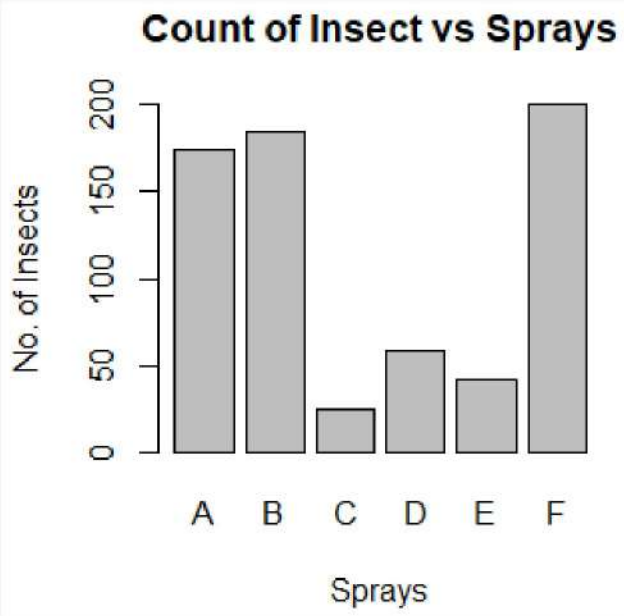
```
> hist(InsectSprays$count,main = "Histogram of Insect Counts",xlab = "Count of Insects",ylab = "Frequency")
```

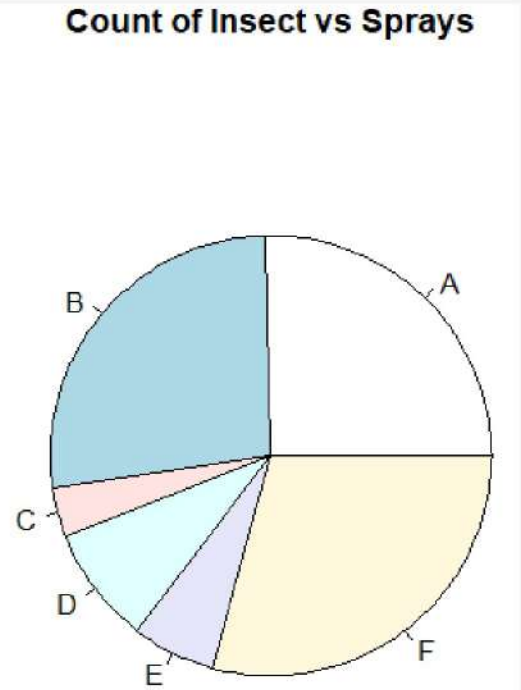


Exercise 14.1.b Obtain the total number of insects found according to each spray . Then, use base R graphics to produce a vertical barplot and a pie chart of these totals, labeling each plot appropriately.

Solution :

```
> t<-tapply(InsectSprays$count, InsectSprays$spray, sum)
> t
  A   B   C   D   E   F
174 184  25  59  42 200
> barplot(t, xlab = "Sprays", ylab = "No. of Insects", main = "Count of
Insect vs Sprays",space=NULL)
> pie(t, main = "Count of Insect vs Sprays")
```

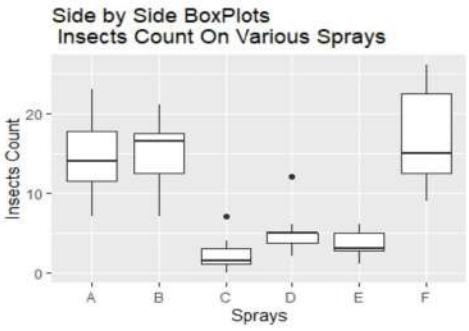




Exercise 14.1.c Use ggplot2 functionality to generate side-by-side boxplots of the counts of insects according to each spray type and include appropriate axis labels and a title.

Solution :

```
>qplot(InsectSprays$spray,InsectSprays$count,geom="boxplot",xlab="Spray
s",ylab="Insects Count",main="Side by Side BoxPlots \n Insects Count On
Various Sprays")
```

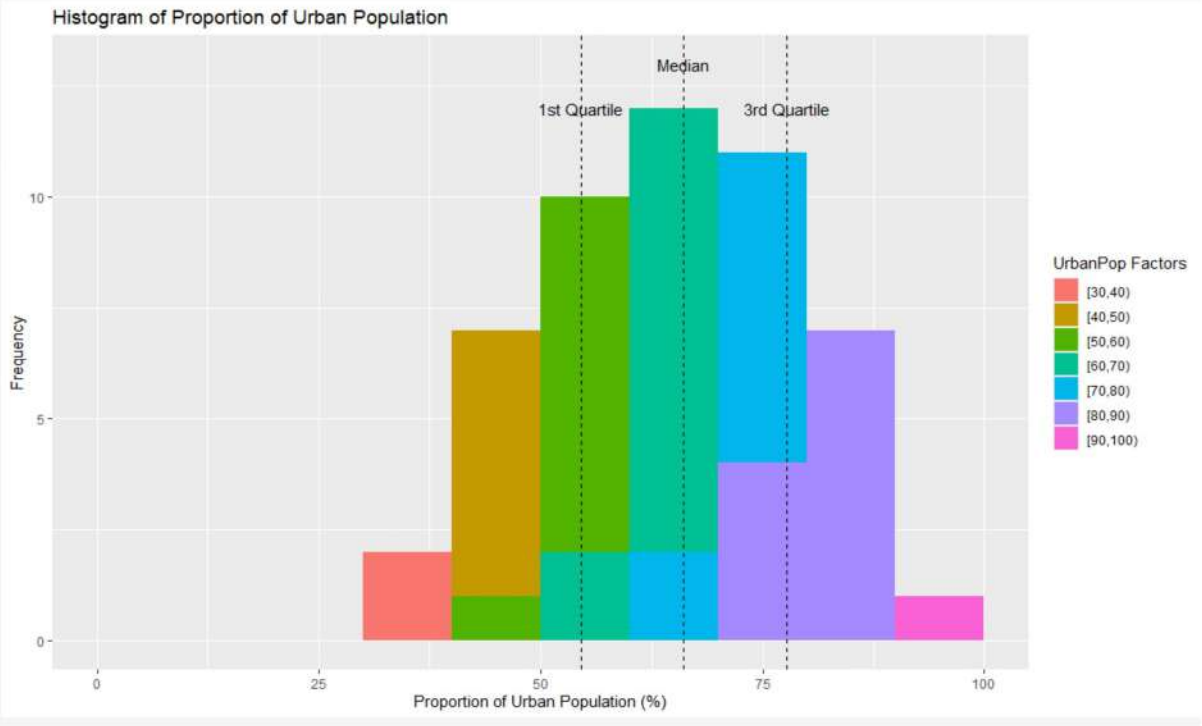


Yet another of R's useful ready-to-use data sets is USArrests, containing data on the number of arrests for murder, rape, and assault per 100,000 individuals in each of the 50 states of the United States, recorded in 1973 (see, for example, McNeil, 1977). It also includes a variable giving the percentage of urban-based population in each state. Briefly inspect the data frame object and the accompanying documentation ?USArrests. Then complete the following:

Exercise 14.1.d Use ggplot2 functionality to generate a right-exclusive histogram of the proportion of urban population for the states. Set your breaks to be 10 units each, between 0 and 100. Have the histogram show the first quartile, the median, and the third quartile; then provide a matching legend. Use colors as you like and include appropriate axis annotation.

Solution :

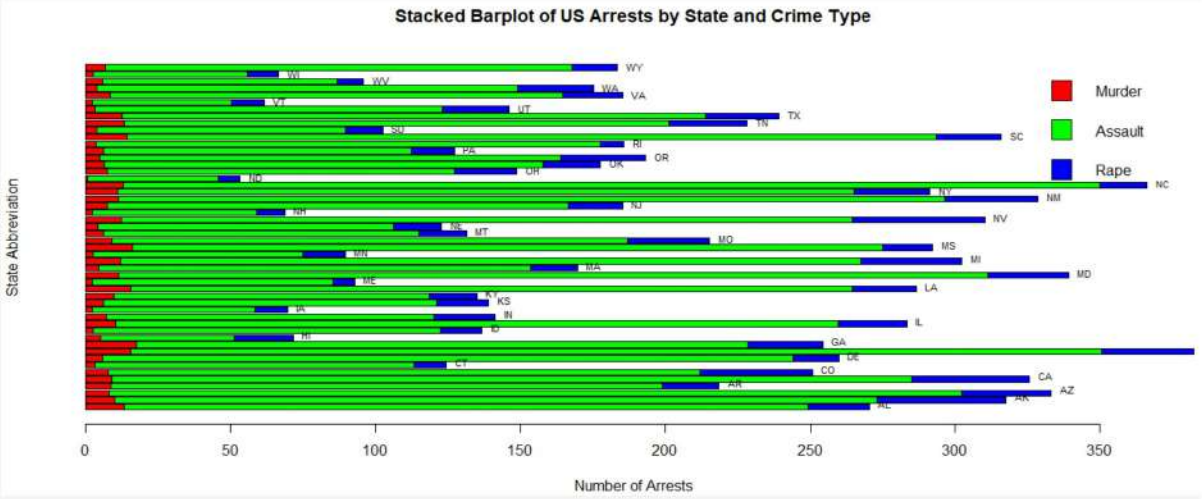
```
xint = quantile(USArrests$UrbanPop, c(0.25, 0.5, 0.75))
ggplot(USArrests,aes(UrbanPop)) +
geom_histogram(breaks = seq(0,100,by=10),aes(fill=factor(cut(UrbanPop,
breaks = seq(0, 100, by = 10), right = FALSE))), show.legend = TRUE)+
geom_vline(xintercept = xint,linetype = "dashed")+
annotate("text", y = 12, x = xint[1], label = "1st Quartile")+
annotate("text", y = 13, x = xint[2], label = "Median")+
annotate("text", y = 12, x = xint[3], label = "3rd Quartile")+
labs(x = "Proportion of Urban Population (%)",y = "Frequency",title =
"Histogram of Proportion of Urban Population")+
scale_fill_discrete(name = "UrbanPop Factors")
```



Exercise 14.1.e The code **t(as.matrix(USArrests[,-3]))** creates a matrix of the USArrests data without the urban population column, and the built-in R object state.abb provides the two-letter state abbreviations, in alphabetical order, as a character vector. Use these two structures and base R graphics to produce a horizontal, stacked barplot with the horizontal bars labeled with state abbreviations and with each bar split according to the type of crime (murder, rape, and assault). Include a legend.

Solution :

```
arrests <- t(as.matrix(USArrests[, -3]))
barplot(arrests,horiz = TRUE,col = c("red", "green", "blue"),# Colors
for murder, rape, and assault
legend.text = TRUE, axisnames = FALSE,args.legend = list(x =
"topright",bty = "n"),main = "Stacked Barplot of US Arrests by State
and Crime Type",xlab = "Number of Arrests",ylab = "State Abbreviation")
text(x = colSums(arrests),y = 1:length(state.abb)*1.19,labels =
state.abb,pos = 4, cex=0.7)
```



Exercise 14.1.f Define a new factor vector urbancat that is set to 1 if the corresponding state has an urban population percentage greater than the median percentage and is set to 0 otherwise.

Solution :

```
> median <- median(USArrests$UrbanPop)
> urbancat <- ifelse(USArrests$UrbanPop > median, 1, 0)
> urbancat <- factor(urbancat, levels = c(0, 1), labels = c("0", "1"))
> urbancat
 [1] 0 0 1 0 1 1 1 1 1 0 1 0 1 0 0 0 0 0 0 1 1 1 0 0 1 0 0 1 0 1 1 1 0
0 1 1 1 1 1 0 0 0 1 1 0 0 1 0 0 0
Levels: 0 1
```

Exercise 14.1.g Create a new copy of USArrests in your workspace, after deleting the UrbanPop column, leaving just

the three crime rate variables. Then insert a new,fourth column in this object with urbancat.

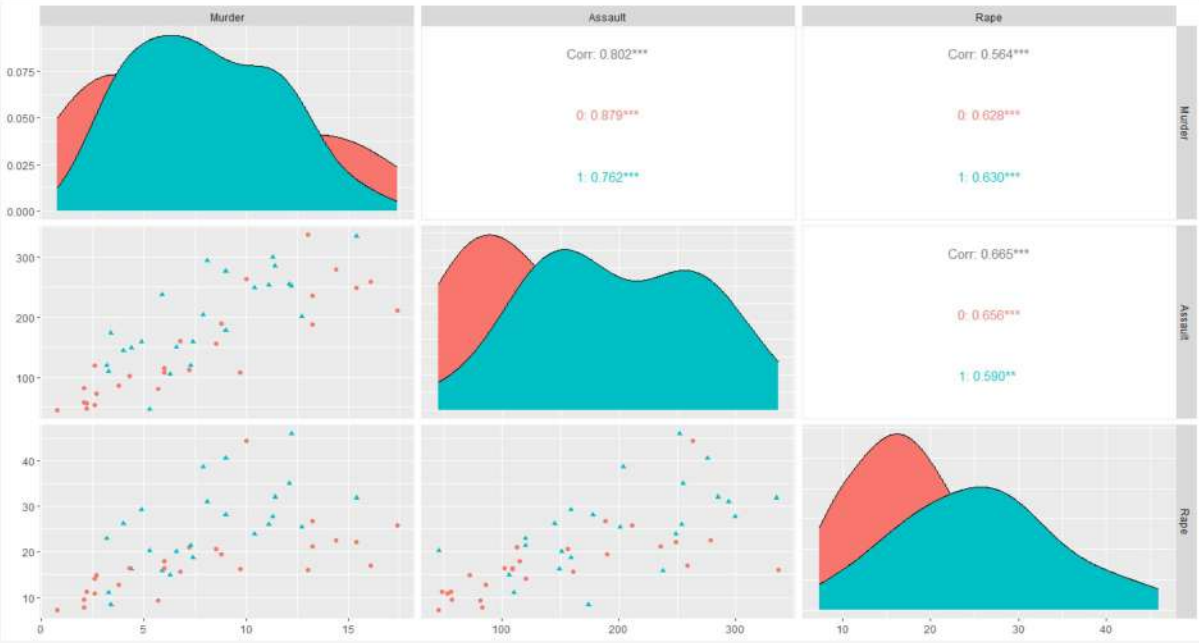
Solution :

```
new_USArrests<-USArrests[,-3]
new_USArrests<-cbind(new_USArrests,urbancat)
```

Exercise 14.1.h Use the data frame from (g) to produce a scatterplot matrix and other associated plots of the three crime rates against one another via GGally functionality. Use shapes to split the crime rates according to the two levels of urbancat.

Solution :

```
library(GGally)
ggpairs(new_USArrests,columns = c("Murder", "Assault", "Rape"),mapping
= aes(col = urbancat, shape = urbancat))
```



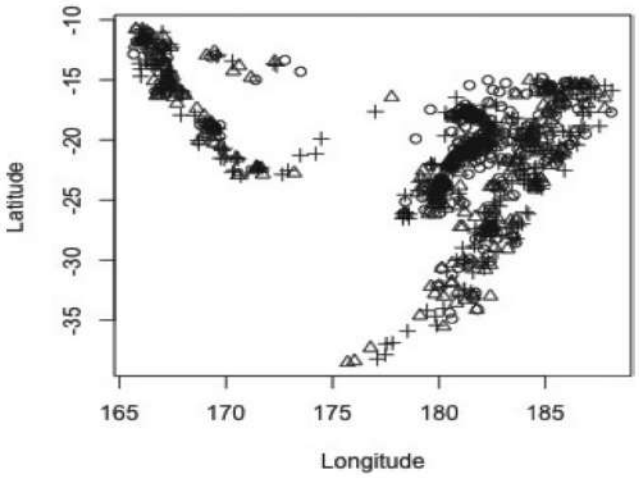
Return to the built-in quakes data set.

Exercise 14.1.i Create a factor vector corresponding to the magnitudes. Each entry should assume one of three categories based on breaks marked by the minimum magnitude,the 1/3th quantile,the 2/3th quantile, and the maximum magnitude.

Solution :

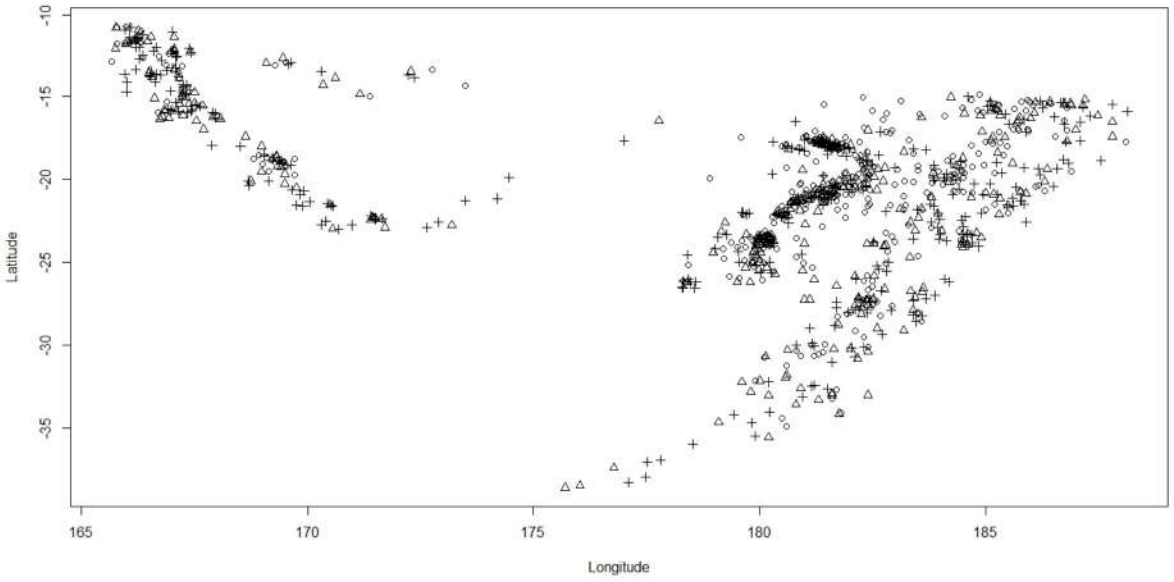
```
> qmag <- quantile(quakes$mag,c(0,1/3,2/3,1))
> magfac <- factor(cut(quakes$mag, breaks = qmag, labels = c(0, 1 ,2),
levels = c("Low", "Medium", "High"),include.lowest = TRUE))
> head(magfac)
[1] 2 0 2 0 0 0
Levels: 0 1 2
```

Exercise 14.1.j Re-create the plot shown next, where low-,medium-, and high magnitude events,according to your factor vector from (i), are plotted with pch being assigned 1, 2, and 3, respectively.



Solution :

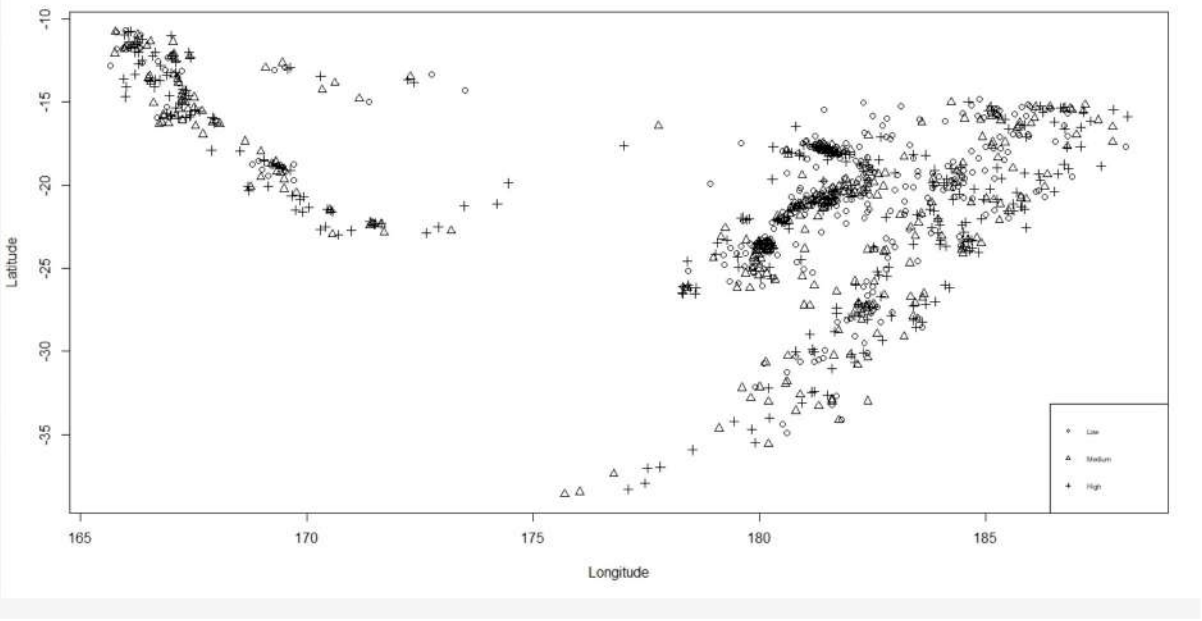
```
quakes<-cbind(quakes,magfac)
plot(quakes[,2],quakes[,1],type="n",xlab="Longitude",ylab="Latitude")
points(quakes[quakes$magfac==0,2],quakes[quakes$magfac==0,1],pch=1)
points(quakes[quakes$magfac==1,2],quakes[quakes$magfac==1,1],pch=2)
points(quakes[quakes$magfac==2,2],quakes[quakes$magfac==2,1],pch=3)
```



Exercise 14.1.k Add a legend to the plot from (j) to reference the three pch values.

Solution :

```
legend("bottomright",legend=c("Low", "Medium",
"High"),pch=C(1:3),cex=0.5)
```



Conclusion -

Studied and Implemented the Various Data Visualisation Plots in R programming language Successfully.

EXPERIMENT-9

Aim - To Study and Implementing the Regression Line(in ScatterPlot), Skewness, Kurtosis in R programming language.

Theory-

Regression Line:
A Regression line in a scatterplot represents the best-fit line through the data points, indicating the relationship between two variables. It is typically calculated using a statistical method such as ordinary least squares regression. In R, you can add a regression line to a scatterplot using functions like lm() to fit a linear model and abline() to plot the regression line.

Skewness:
Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. A distribution is positively skewed if the tail on the right side of the distribution is longer or fatter than the left side, and vice versa for negative skewness.

Kurtosis:
Kurtosis is a measure of the "tailedness" of the probability distribution of a real-valued random variable. It describes the peakedness or flatness of the distribution. A distribution with positive kurtosis (leptokurtic) has a sharper peak and heavier tails, while a distribution with negative kurtosis (platykurtic) has a flatter peak and lighter tails.

Syntax:

```
>plot(df$x, df$y)                    # Create a Scatterplot
> abline( lm(y ~ x, data = df) )     # Add regression line
abline(linear model)
> library(e1071)
> skewness(df$x)                  # Calculate Skewness
> kurtosis(df$x)                    # Calculate Kurtosis
```

EXAMPLES

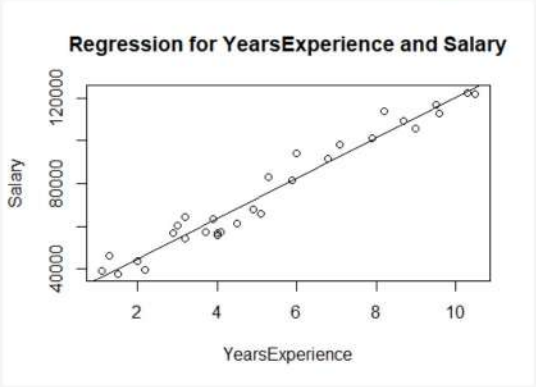
Example 1: Create a Scatterplot with a Regression Line in R using dataset : Salary_Data.xls.

Source:https://www.geeksforgeeks.org/how-to-create-a-scatterplot-with-a-regression-line-in-r/

Solution :

```
> file.choose()
[1] "C:\\Users\\Microsoft\\Downloads\\Salary_Data.xls"
> library(readxl)
> Salary_Data <- read_excel("Salary_Data.xls")
> plot(Salary_Data$YearsExperience,Salary_Data$Salary,
+      main='Regression for YearsExperience and Salary',
```

```
+      xlab='YearsExperience',ylab='Salary')
> abline(lm(Salary~YearsExperience,data=Salary_Data))
```



Example 2: Using same dataset,Calculate the Skewness of all Variables in it.

Solution :

```
> library(e1071)
> skewness(Salary_Data$YearsExperience)
[1] 0.3424477
> skewness(Salary_Data$Salary)
[1] 0.3194946
```

Example 3: Using same dataset,Calculate the Kurtosis of all Variables in it.

Solution :

```
> library(e1071)
> kurtosis(Salary_Data$YearsExperience)
[1] -1.17293
> kurtosis(Salary_Data$Salary)
[1] -1.395477
```

Conclusion -

Studied and Implemented the Regression Line(in ScatterPlot), Skewness, Kurtosis in R programming language Successfully.

EXPERIMENT-10

Aim - To Study and Implement the Probability Distribution and Random Variable in R programming language.

Theory-

Probability: The likelihood of an event occurring, measured between 0 (impossible) and 1 (certain).
Event: An outcome or set of outcomes of an experiment or random phenomenon.
Conditional Probability: The probability of an event occurring given that another event has already occurred.
Intersection: The set of outcomes that are common to two or more events.
Union: The set of outcomes that belong to either one or both of two or more events.
Complement: The set of outcomes that do not belong to a given event.

Random Variable: A variable whose possible values are outcomes of a random phenomenon. It can be Discrete or Continuous.
Realization: The specific value that a random variable takes on in an experiment.
Probability Distribution: A function that assigns probabilities to the outcomes of a random variable.
Cumulative Distribution: The accumulation of probabilities up to a certain point in a probability distribution.

EXERCISES

You have a standard deck of 52 playing cards. There are two colors (black and red) and four suits (spades are black, clubs are black, hearts are red, and diamonds are red). Each suit has 13 cards, in which there is an ace, numbered cards from 2 to 10, and three face cards (jack, queen, and king) .

Exercise 15.1.a You randomly draw and then replace a card. What's the probability it's an ace? What's the probability it's the 4 of spades?
Solution :

```
Pr(ace) = 4/52 = 1/13
Pr(4 of Spades) = 1/52
```

Exercise 15.1.b You randomly draw a card, and after replacing it, you draw another. Let *A* be the event that the card is a club; let *B* be the event that the card is red. What is Pr(A|B) ? That is, what is the probability the second card is a club *given* the first one was a red card? Are the two events independent?
Solution :

```
Pr(A) = 13/52 = 0.25
Pr(B) = 26/52 =0.5
```

Pr(A|B) = 13/52 = 0.25

Since Pr(A|B) = Pr(A) →Two events are independent

Exercise 15.1.c Repeat (b) , this time assuming that when the first (club) card is drawn, it is not replaced. Would this change your answer to (b) in terms of independence?

Solution :

Pr(A|B) = 26/51

Since Pr(A|B) != Pr(A) →Two events are NOT independent

Exercise 15.1.d Let C be the event a card is a face card, and let D be the event a card is black. You draw a single card. Evaluate Pr(C⋂D). Are the two events mutually exclusive?

Solution :

Pr(C) = 12/52 = 3/13

Pr(D) = 26/52 = 0.5

Pr(C⋂D) = 6/52

Since Pr(C⋂D) != 0 →Two events are NOT Mutually Exclusive

Exercise 15.2.a For each of the following definitions, identify whether it's best described as a random variable or as a *realization* of a random variable. Furthermore, identify whether each statement describes a continuous or a discrete quantity.

1. The number of coffees x made by your local shop on June 3, 2016
2. The number of coffees X made by your local shop on any given day
3. Y, whether or not it rains tomorrow
4. Z, the amount of rain that falls tomorrow
5. How many crumbs k on your desk right now
6. Total collective weight W of the crumbs on your desk at any specified time

Solution :

1. Random variable: X — Continuous quantity
2. Random variable: X — Discrete quantity
3. Random variable: Y — Discrete quantity
4. Random variable: Z — Continuous quantity
5. Realization of a random variable: k — Discrete quantity
6. Random variable: W — Continuous quantity

Exercise 15.2.b Suppose you construct the following table providing probabilities associated with the random variable S, the total stars given to any movie in a particular genre by a certain critic:

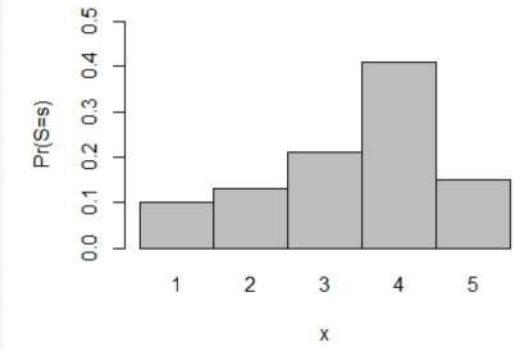
s	1	2	3	4	5
$Pr(S = s)$	0.10	0.13	0.21	???	0.15

1. Assuming this table describes the complete set of outcomes, evaluate the missing probability Pr(S = 4).
2. Obtain the cumulative probabilities.
3. What is the mean of S, the expected number of stars this critic will award any given movie in this genre?

4. What is the standard deviation of *S*?
5. What is the probability that any given movie in this genre will be given at least three stars?
6. Visualize, and briefly comment on the appearance of, the probability mass function.

Solution :

```
Pr(S=4) = 1 - 0.10 - 0.13 - 0.21 - 0.15 = 0.41
> S<- c(1:5)
> S.prob<- c(0.10,0.13,0.21,0.41,0.15)
> S.cumul <- cumsum(S.prob)
> S.cumul
[1] 0.10 0.23 0.44 0.85 1.00
> S.mean<-sum(S*S.prob)
> S.mean
[1] 3.38
> S.var<-sum((S-S.mean)^2*S.prob)
>sqrt(S.var)  # Std Daviation
[1] 1.181355
> 1-S.cumul[2]
[1] 0.77
>
barplot(S.prob,ylim=c(0,0.5),names.arg=S,space=0,xlab="x",ylab="Pr(S=s)
")
# 77% movies get good rating(3 or more) by this critic
```



Exercise 15.2.c Return to the picnic temperature example based on the random variable *W* defined in Section 15.2.3.

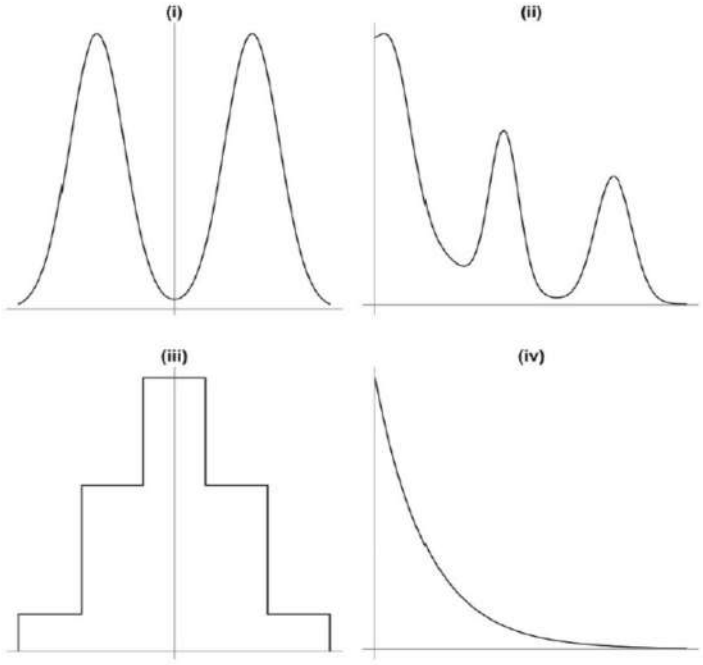
1. Write an R function to return *f(w) (Discrete)* as per Equation (15.5) for any numeric vector of values supplied as *w*. Try to avoid using a loop in favor of vector-oriented operations.
2. Write an R function to return *F(w) (continuous)* as per Equation (15.6) for any numeric vector of values supplied as *w*. Again, try to avoid using a loop, either explicit or implicit.
3. Use your functions from (i) and (ii) to confirm the results from the text, in other words, that *f* (55.2) = 0.02432 and that F(55.2) = 0.184832.
4. Make use of your function for *F(w)* to compute *Pr(W* > 60). Hint: Note that because the total area underneath *f(w)* is one, *Pr(W* >60) = 1-*Pr(W*<=60).
5. Find Pr(60.3 < *W* < 76.89).

Solution :

```
> w<-seq(35,95,5)
> funcw <-function(w){
+   lower.w <- w>=40 & w<=65
+   upper.w <- w>65 & w<=90
+   fw <- rep(0,length(w))
+   fw[lower.w] <- (w[lower.w]-40)/625
```

```
+   fw[upper.w] <- (90-w[upper.w])/625
+   return(fw)
+ }
> fw<-funcw(w)
> fw
 [1] 0.000 0.000 0.008 0.016 0.024 0.032 0.040 0.032 0.024 0.016 0.008 0.000
0.000
> Funcw <-function(w){
+   lower.w <- w>=40 & w<=65
+   upper.w <- w>65 & w<=90
+   Fw <- rep(0,length(w))
+   Fw[lower.w] <- (w[lower.w]^2-80*w[lower.w]+1600)/1250
+   Fw[upper.w] <- (180*w[upper.w]-w[upper.w]^2-6850)/1250
+   Fw[w>90] <- 1
+   return(Fw)
+ }
> Fw<-Funcw(w)
> Fw
 [1] 0.00 0.00 0.02 0.08 0.18 0.32 0.50 0.68 0.82 0.92 0.98 1.00 1.00
> funcw(55.2)
[1] 0.02432
> Funcw(55.2)
[1] 0.184832
> 1-Funcw(60)
[1] 0.68
> Funcw(76.89)-Funcw(60.3)
[1] 0.5328303
```

Exercise 15.2.d Assume each of the following plots labeled (i)-(iv) shows the general appearance of a probability distribution. Use terminology from Section 15.2.4 to describe the shape of each.



Solution :

(i) Symmetric	Non Skewed	Bimodal
(ii) Asymmetric	Right/Positive Skew	Trimodal
(iii) Symmetric	Non Skewed	Unimodal
(iv) Asymmetric	Right/Positive Skew	Unimodal

Conclusion -

Studied and Implemented the Probability Distribution and Random Variable in R programming language Successfully.

EXPERIMENT-11

Aim - To Study and Implement the Various Probability Distributions and Density Functions in R programming language.

Theory-

Probability Distribution Functions:

d <Distribution> : Probability Mass/Density function .
p <Distribution> : Cumulative distribution/Left Probabilities function .
q <Distribution> : Quantile function (inverse of p <Distribution>).
r <Distribution> : Random variate generation function.
Where <Distribution> can be **binom pois unif norm exp**

Probability Mass Function (PMF):

-A function that gives the probability that a discrete random variable is exactly equal to some value.

1. Binomial Distribution: (dbinom pbinom qbinom rbinom)
- Models the number of successes in a fixed number of independent Bernoulli trials, where each trial has the same probability of success.
- It's characterized by two parameters: the number of trials (n) and the probability of success (p) in each trial.

2. Poisson Distribution: (dpois ppois qpois rpois)
- Models the number of events occurring in a fixed interval of time or space, given a constant average rate of occurrence.
- It's often used to describe the number of arrivals in a fixed period of time or the number of defects in a fixed area.

Probability Density Function (PDF):

- A function that describes the likelihood of a continuous random variable falling within a particular range of values.

3. Uniform Distribution: (dunif punif qunif runif)
- Describes a continuous random variable where all intervals of the same length have an equal probability of being observed.
- It's characterized by two parameters: the minimum and maximum values of the interval.

4. Normal Distribution: (dnorm pnorm qnorm rnorm)
- One of the most commonly used distributions in statistics, describing a continuous random variable with a symmetric bell-shaped curve.

- It's characterized by two parameters: the mean (mu) and the standard deviation (sigma).

5. Exponential Distribution: (dexp pexp qexp rexp)
- Models the time between events in a Poisson process, where events occur continuously and independently at a constant average rate over time.
- It's often used to describe the lifetimes of objects or the time between arrivals in a queuing system.
- It's characterized by one parameter: the rate (lambda), which is the average number of events occurring in a unit of time.

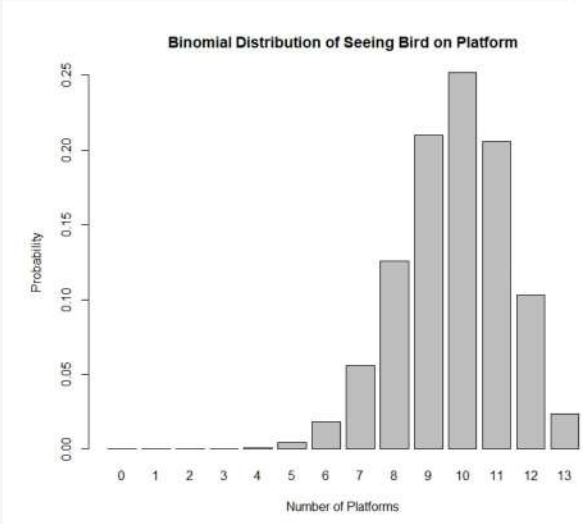
EXERCISES

A forested nature reserve has 13 bird-viewing platforms scattered throughout a large block of land. The naturalists claim that at any point in time, there is a 75 percent chance of seeing birds at each platform. Suppose you walk through the reserve and visit every platform. If you assume that all relevant conditions are satisfied, let X be a binomial random variable representing the total number of platforms at which you see birds.

Exercise 16.1.a Visualize the probability mass function of the binomial distribution of interest.

Solution :

```
> bird.prob <- dbinom(0:13,size=13,prob=0.75)#a
> bird.prob
 [1] 1.490116e-08 5.811453e-07
1.046062e-05 1.150668e-04
8.630008e-04
 [6] 4.660204e-03 1.864082e-02
5.592245e-02 1.258255e-01
2.097092e-01
[11] 2.516510e-01 2.058963e-01
1.029481e-01 2.375726e-02
> barplot(bird.prob,names.arg = 0:13,
xlab = "Number of Platforms", ylab =
"Probability", main = "Binomial
Distribution of Seeing Bird on
Platform")
```



Exercise 16.1.b What is the probability you see birds at all sites?

Solution :

```
> dbinom(13, size = 13, prob = 0.75)#b
[1] 0.02375726
```

Exercise 16.1.c What is the probability you see birds at more than 9 platforms?

Solution :

```
> sum(dbinom(10:13,size=13,prob=0.75))#c
[1] 0.5842527
```

Exercise 16.1.d What is the probability of seeing birds at between 8 and 11 platforms (inclusive)? Confirm your answer by using only the d-function and then again using only the p-function.

Solution :

```
> sum(dbinom(8:11,size=13,prob=0.75))#d
[1] 0.793082
> pbinom(11,size=13,prob=0.75)-pbinom(7,size=13,prob=0.75)
[1] 0.793082
```

Exercise 16.1.e Say that, before your visit, you decide that if you see birds at fewer than 9 sites, you'll make a scene and demand your entry fee back. What's the probability of embarrassing yourself in this way?

Solution :

```
> pbinom(8,size=13,prob=0.75)#e
[1] 0.2060381
```

Exercise 16.1.f Simulate realizations of X that represent 10 different visits to the reserve; store your resulting vector as an object.

Solution :

```
> rbinom(10,size=13,prob=0.75)#f
 [1]  8  8  9  7  8 10  8 10 10 11
```

Exercise 16.1.g Compute the mean and standard deviation of the distribution of interest.

Solution :

```
> 13*0.75  #g
[1] 9.75
> sqrt(13*0.75*0.25)
[1] 1.561249
```

→Every Saturday, at the same time, an individual stands by the side of a road and tallies the number of cars going by within a 120-minute window. Based on previous knowledge, she believes that the mean number of cars going by during this time is exactly 107. Let X represent the appropriate Poisson random variable of the number of cars passing her position in each Saturday session.

Exercise 16.2.a What is the probability that more than 100 cars pass her on any given Saturday?

Solution :

```
> 1-ppois(q=100,lambda=107)
[1] 0.7319128
```

Exercise 16.2.b Determine the probability that no cars pass.

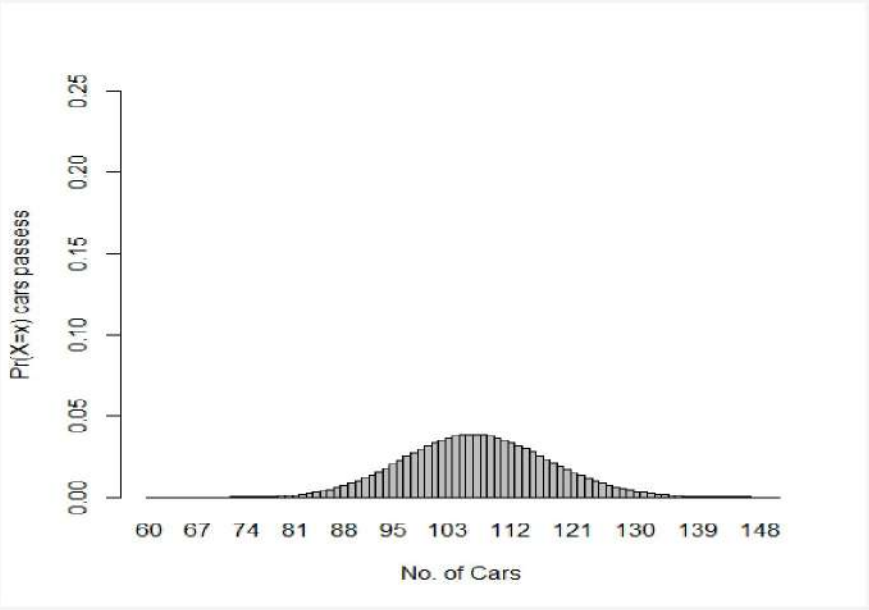
Solution :

```
> dpois(x=0,107)
[1] 3.39227e-47
```

Exercise 16.2.c Plot the relevant Poisson mass function over the values in 60 <= x <= 150.

Solution :

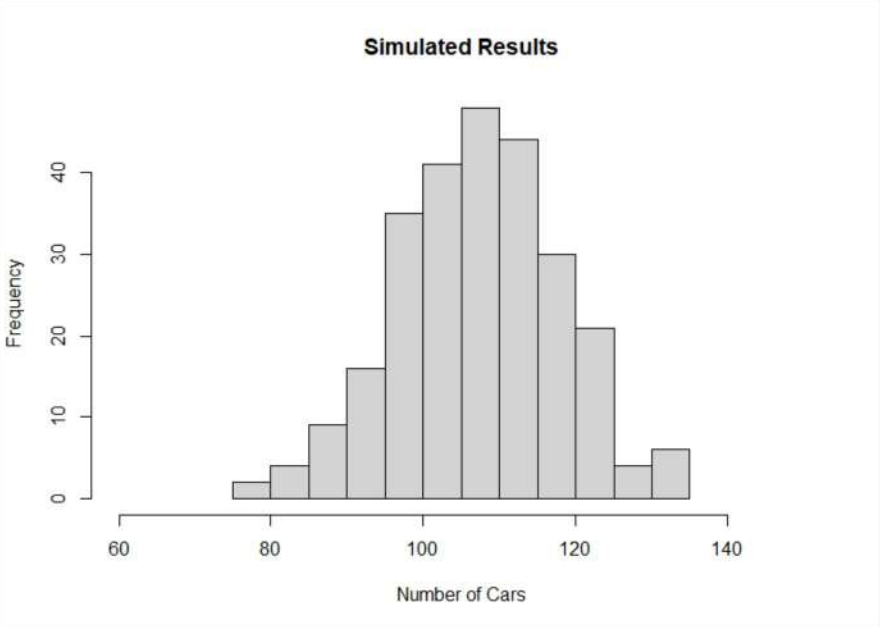
```
>
barplot(dpois(60:150,107),ylim=c(0,0.25),names.arg=60:150,space=0,xlab=
"No. of Cars",ylab="Pr(X=x) cars passess")
```



Exercise 16.2.d Simulate 260 results from this distribution (about five years of weekly Saturday monitoring sessions). Plot the simulated results using hist; use xlim to set the horizontal limits from 60 to 150. Compare your histogram to the shape of your mass function from (c).

Solution :

```
> hist(rpois(260,107),xlim = c(60, 150), main = "Simulated Results",
xlab = "Number of Cars", ylab = "Frequency")
```



→You visit a national park and are informed that the height of a certain species of tree found in the forest is uniformly distributed between 3 and 70 feet.

Exercise 16.3.a What is the probability you encounter a tree shorter than 5.5 feet?

Solution :

```
> punif(q=5.5,min=3,max=70)
[1] 0.03731343
```

Exercise 16.3.b For this probability density function, what is the height that marks the cutoff point of the tallest 15 percent of trees?

Solution :

```
> qunif(p=1-0.15,min=3,max=70)
[1] 59.95
```

Exercise 16.3.c Evaluate the mean and standard deviation of the tree height distribution.

Solution :

```
> m<-(3+70)/2   #mean
> m
[1] 36.5
> sd<-sqrt( (70-3)^2/12 ) #std Daviation
> sd
[1] 19.34123
```

Exercise 16.3.d Using (c), confirm that the chance that you encounter a tree with a height that is within half a standard deviation (that is, below or above) of the mean height is roughly 28.9 percent.

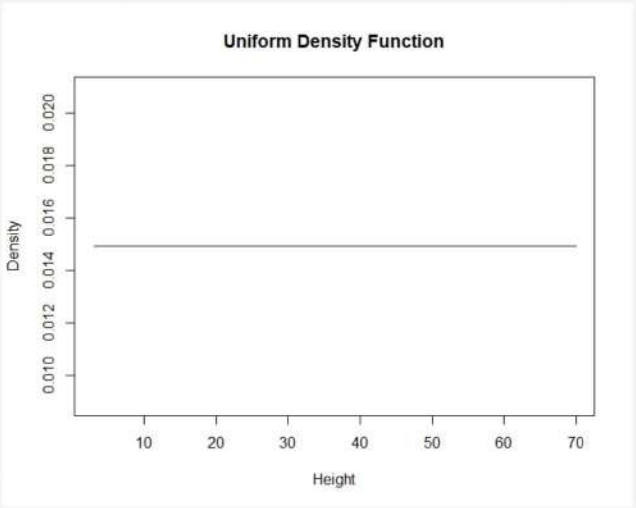
Solution :

```
> punif(m+0.5*sd,3,70)-punif(m-0.5*sd,3,70)
[1] 0.2886751
```

Exercise 16.3.e At what height is the density function itself? Show it in a plot.

Solution :

```
> 1/(70-3)
[1] 0.01492537
> curve(dunif(x, min = 3, max = 70), from = 3, to = 70, xlab =
"Height", ylab = "Density", main = "Uniform Density Function")
```



Exercise 16.3.f Simulate 10 observed tree heights. Based on these data, use quantile (refer to Section 13.2.3) to estimate the answer you arrived at in (b). Repeat your simulation, this time generating 1,000 variates, and estimate (b) again. Do this a handful of times, taking a mental note of your two estimates each time. Overall, what do you notice of your two estimates (one based on 10 variates at a time and the other based on 1,000) with respect to the "true" value in (b)?

Solution :

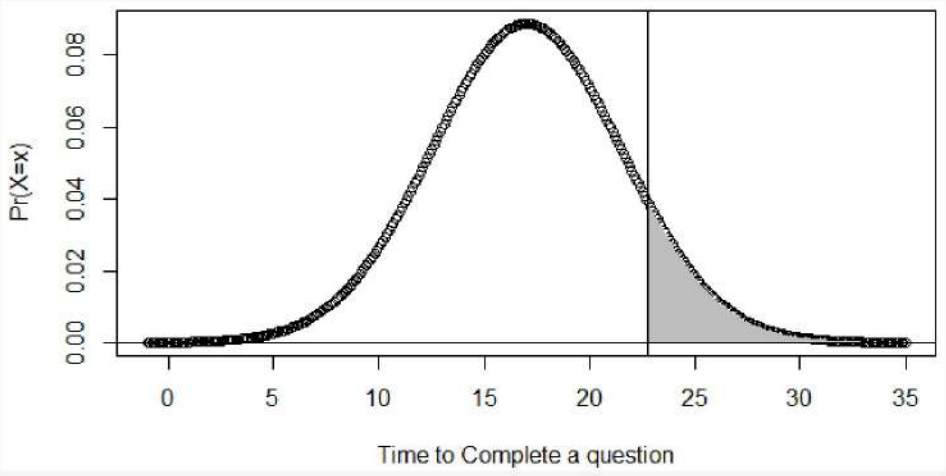
```
> quantile( runif(10,min=3,max=70),0.85 )
     85%
48.99629
> quantile( runif(10,min=3,max=70),0.85 )
     85%
46.42475
> quantile( runif(1000,min=3,max=70),0.85 )
     85%
59.87198
> quantile( runif(1000,min=3,max=70),0.85 )
     85%
59.11549
# We Noticed On Increasing the Sample Size,The Value is more Closer to
Actual value in (b),that is 59.95
```

Exercise 16.4.a A tutor knows that the length of time taken to complete a certain statistics question by first-year undergraduate students, X , is normally distributed with a mean of 17 minutes and a standard deviation of 4.5 minutes.

1. What is the probability a randomly selected undergraduate takes more than 20 minutes to complete the question?
2. What's the chance that a student takes between 5 and 10 minutes to finish the question?
3. Find the time that marks off the slowest 10 percent of students.
4. Plot the normal distribution of interest between $\pm 4\sigma$ and shade in the probability area of (iii), the slowest 10 percent of students.
5. Generate a realization of times based on a class of 10 students completing the question.

Solution :

```
> 1-pnorm(20,mean=17,sd=4.5) #1
[1] 0.2524925
> pnorm(10,mean=17,sd=4.5)-pnorm(5,mean=17,sd=4.5) #2
[1] 0.05607653
> qnorm(1-0.1,17,4.5) #3
[1] 22.76698
> x<-seq(-1,35,0.1) #4
> fx<-dnorm(x,17,4.5)
> plot(x,fx,type="p",xlim=c(17-4*4.5 , 17+4*4.5),xlab="Time to Complete
a question",ylab="Pr(X=x)")
> abline(h=0)
> abline(v=qnorm(1-0.1,17,4.5))
> x_ = x[x>=qnorm(1-0.1,17,4.5)]
> fx_=fx[x>=qnorm(1-0.1,17,4.5)]
>polygon(rbind(c(qnorm(1-0.1,17,4.5),0),cbind(x_,fx_)),border=NA,col="g
ray")
> rnorm(10,17,4.5) #5
 [1] 16.359180 17.677361  9.618489 16.757467 16.170032 14.910986
19.520354
 [8] 13.583234 14.473402 19.519251
```



Exercise 16.4.b A meticulous gardener is interested in the length of blades of grass on his lawn. He believes that blade

length X follows a normal distribution centered on 10 mm with a variance of 2 mm.
i. Find the probability that a blade of grass is between 9.5 and 11 mm long.
ii. What are the standardized values of 9.5 and 11 in the context of this distribution? Using the standardized values, confirm that you can obtain the same probability you found in (i) with the standard normal density.
iii. Below which value are the shortest 2.5 percent of blade lengths found?
iv. Standardize your answer from (iii).

Solution :

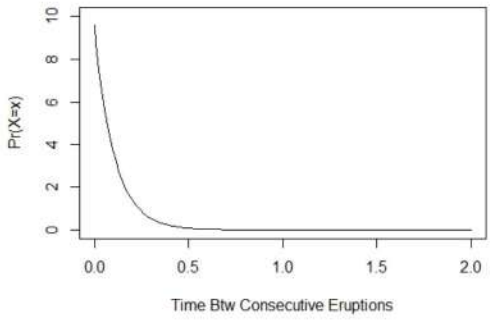
```
> pnorm(q=11,mean=10,sd=sqrt(2))-pnorm(9.5,10,sqrt(2)) # i
[1] 0.3984131
> z_9.5 <- (9.5 - 10) / sqrt(2) # ii
> z_9.5
[1] -0.3535534
> z_11 <- (11 - 10) / sqrt(2)
> z_11
[1] 0.7071068
> pnorm(z_11) - pnorm(z_9.5)
[1] 0.3984131
> qnorm(p=0.025,10,sqrt(2)) # iii
[1] 7.228192
> (qnorm(p=0.025,10,sqrt(2))-10)/sqrt(2) # iv
[1] -1.959964
> qnorm(0.025)
[1] -1.959964
```

Exercise 16.5.a Situated in the central north island of New Zealand, the Pohutu geyser is said to be the largest active geyser in the southern hemisphere. Suppose that it erupts an average of 3,500 times every year.
i. With the intention of modeling a random variable X as the time between consecutive eruptions, evaluate the parameter value λe with respect to a time scale in days (assume 365.25 days per year to account for leap years).
ii. Plot the density function of interest. What's the mean wait in days between eruptions?
iii. What's the probability of waiting less than 30 minutes for the next eruption?
iv. What waiting time defines the longest 10 percent of waits? Convert your answer to hours.

Solution :

```
> lambda.e <- 3500/365.25  #i
> lambda.e
[1] 9.582478
```

```
> x<-seq(0,2,length=200) #ii
> fx<-dexp(x,rate=lambda.e)
> plot(x,fx,xlim=c(0,2),ylim=c(0,10),type="l",xlab="Time Btw
Consecutive Eruptions",ylab="Pr(X=x)")
> 1/lambda.e  #mean
[1] 0.1043571
> pexp(q=(30/(60*24)) ,lambda.e) #iii
[1] 0.1809703
> qexp(1-0.1 ,lambda.e)    #iv
[1] 0.2402912
> qexp(1-0.1 ,lambda.e)*24
[1] 5.766989
```



Exercise 16.5.b You can also use the exponential distribution to model certain product survival times, or "time-to-failure" type of variables. Say a manufacturer of a particular air conditioning unit knows that the product has an average life of 11 years before it needs any type of repair callout. Let the random variable X represent the time until the necessary repair of one of these units and assume X follows an exponential distribution with $\lambda e = 1/11$.

i. The company offers a five-year full repair warranty on this unit. What's the probability that a randomly selected air conditioner owner makes use of the warranty?

ii. A rival company offers a six-year guarantee on its competing air conditioning unit but knows that its units last, on average, only nine years before requiring some kind of repair. What are the chances of making use of that warranty?

iii. Determine the probabilities that the units in (i) and the units in (ii) last more than 15 years.

Solution :

```
> pexp(5,rate=1/11) #i
[1] 0.3652636
> pexp(6,1/9)       #ii
[1] 0.4865829
> 1-pexp(15,1/11)  #iii
[1] 0.2557292
> 1-pexp(15,1/9)
[1] 0.1888756
```

Conclusion -

Studied and Implemented the Various Probability Distribution and Density Functions in R programming language Successfully.

EXPERIMENT-12

Aim - To Study and Implement the Hypothesis Testing (1 Sample & 2 Sample) in R programming language.

Theory-

Hypothesis Testing:

Hypothesis testing is a statistical method used to make inferences about a population based on sample data. There are various types of hypothesis testing, including one-sample and two-sample tests.

One Sample Hypothesis Testing:

- In one-sample hypothesis testing, we compare a sample statistic (such as the sample mean) to a known population parameter or a hypothesized value.
- The process typically involves defining a null hypothesis (H0) and an alternative hypothesis (Ha), then using sample data to assess the likelihood of observing the sample statistic under the null hypothesis.
- Common tests in one-sample hypothesis testing include the one-sample t-test for means and the one-sample z-test for proportions.

1. Tests about a Proportion using the test statistic (pnorm and qnorm):

- Utilizes the normal distribution to assess hypotheses about a population proportion using the standard normal test statistic, often z-score.
- pnorm calculates the cumulative probability associated with a given z-score, while qnorm calculates the z-score corresponding to a specified cumulative probability.

Example :

```
> p <- 0.6  # Sample proportion
> n <- 100  # Sample size
> s <- 0.05 # Significance level
> z <- (p - 0.5) / sqrt(0.5 * (1 - 0.5) / n)  #  z-score
> qnorm(s)#Critical values Left tailed
[1] -1.644854
> qnorm(1-s)            #Right Tailed
[1] 1.644854
> qnorm(1-s/2)        #Two Sided tailed
[1] 1.959964
> pnorm(z) #left tailed p value
[1] 0.9772499
> pnorm(z,lower.tail = FALSE) #Right Tailed p value
[1] 0.02275013
> 2 * pnorm(abs(z), lower.tail = FALSE)  #  two-tailed p-value
```

```
[1] 0.04550026
```

2. Tests about a Proportion using x and n (prop.test):

- prop.test evaluates hypotheses about a population proportion using sample proportions and sample sizes, providing test statistics and p-values based on the binomial distribution.
- It assesses whether the observed proportion significantly differs from a hypothesized proportion, considering sample variability.

Example : A coin is flipped 100 times and comes up heads 43 times. Test the claim that this is a fair coin. Use a 0.05 significance level. Everything in red is typed by the user. Everything in blue is output to the console.

Solution :

```
> successes <- 43  # Number of successes
> prop.test(successes,n=100,p = 0.3,correct=FALSE)

     1-sample proportions test without continuity correction

data:  successes out of 100, null probability 0.3
X-squared = 8.0476, df = 1, p-value = 0.004556
alternative hypothesis: true p is not equal to 0.3
95 percent confidence interval:
 0.3373330 0.5278461
sample estimates:
   p
0.43
```

3. Tests about a mean (sigma unknown) using the test statistic (pt and qt):

- Employed when the population standard deviation is unknown, the t-distribution is utilized to evaluate hypotheses about a population mean using the t-test statistic.
- pt calculates the cumulative probability associated with a given t-score, while qt calculates the t-score corresponding to a specified cumulative probability.

Example :

```
>  data <- c(18, 21, 22, 19, 20, 24, 16, 23, 20, 17)  # Sample data
>  n<-length(data)
>  t <- (mean(data) - 20) / (sd(data) / sqrt(n))#t-score
>  s <- 0.05 # Significance level
>  qt(s,df=n-1)
[1] -1.833113
>  qt(1-s,df=n-1)
[1] 1.833113
>  qt(1-s/2,df=n-1)
[1] 2.262157
>  pt(t,df=n-1)#Left Tailed
[1] 0.5
>  pt(t,df=n-1,lower.tail=FALSE)#Right Tailed
[1] 0.5
```

```
>  2*pt(abs(t),df=n-1,lower.tail = FALSE)#two-tailed
[1] 1
```

4. Tests about a mean (sigma unknown) - From Raw Data. (t.test):

- t.test performs hypothesis tests about a population mean from raw data, utilizing sample statistics to calculate the t-test statistic and its associated p-value.
- This method is applicable when the population standard deviation is unknown and the sample size is relatively small.

Example :

```
> #Testing if the mean height of a sample of students is significantly
different from 170 cm
> h <- c(165, 168, 172, 170, 175, 171, 169, 174, 172, 168)  # Sample heights
> t.test(h, mu = 170)
     One Sample t-test
data:  h
t = 0.41804, df = 9, p-value = 0.6857
alternative hypothesis: true mean is not equal to 170
95 percent confidence interval:
 168.2355 172.5645
sample estimates:
mean of x
    170.4
> t.test(h, mu = 170,altenative="greater")
     One Sample t-test
data:  h
t = 0.41804, df = 9, p-value = 0.6857
alternative hypothesis: true mean is not equal to 170
95 percent confidence interval:
 168.2355 172.5645
sample estimates:
mean of x
    170.4
> t.test(h, mu = 170,altenative="less")
     One Sample t-test
data:  h
t = 0.41804, df = 9, p-value = 0.6857
alternative hypothesis: true mean is not equal to 170
95 percent confidence interval:
 168.2355 172.5645
sample estimates:
mean of x
    170.4
```

5. Tests about a mean (sigma known) using the test statistic (pnorm and qnorm):

- Utilizes the normal distribution to evaluate hypotheses about a population mean when the population standard deviation is known, employing the z-test statistic.

- pnorm calculates the cumulative probability associated with a given z-score, while qnorm calculates the z-score corresponding to a specified cumulative probability.

Example :

```
> # Testing if the mean weight of a sample of 50 individuals is
significantly different from 70 kg,sd=5 kg
> mean_weight <- 68  # Sample mean weight
> z <- (mean_weight - 70) / (5 / sqrt(n))  # z-score
> qnorm(s)
[1] -1.644854
> qnorm(1-s)
[1] 1.644854
> qnorm(1-s/2)
[1] 1.959964
> pnorm(z)
[1] 0.1029516
> pnorm(z,lower.tail = FALSE)
[1] 0.8970484
> 2 * pnorm(abs(z), lower.tail = FALSE)  # two-tailed
[1] 0.2059032
```

Two Sample Hypothesis Testing:

- In two-sample hypothesis testing, we compare two independent samples to determine whether they come from populations with the same distribution or if there is a significant difference between them.
- Like one-sample testing, we formulate null and alternative hypotheses and use sample data to evaluate the evidence against the null hypothesis.
- Common tests in two-sample hypothesis testing include the independent samples t-test for means and the chi-square test for independence for categorical data.

1 Means - using raw data:

a) Hypothesis Tests for Mean Differences: Paired Data (t.test):

Assesses whether the mean difference between paired observations is statistically significant, commonly used in before-after studies or matched pairs.

Example : Test the claim that, on average, the drug lowers cholesterol in all men. I.e., test the claim that d > 0. Test this at the 0.05 significance level.

Solution :

```
> before <- c(237,289,257,228, 303, 275, 262,304,244,233)
> after <- c(194, 240, 230,186,265,222,242,281,240,212)
> t.test (before,after,paired=TRUE, alternative="greater", mu=0)

     Paired t-test

data:  before and after
t = 6.5594, df = 9, p-value = 5.202e-05
alternative hypothesis: true mean difference is greater than 0
95 percent confidence interval:
 23.05711      Inf
```

```
sample estimates:
mean difference
             32
```

b) Hypothesis Tests for Two Means: Independent Data (t.test):

Determines whether the difference between the means of two independent groups is statistically significant, assuming equal or unequal variances based on the sample data.

Example : Test the claim that the mean cholesterol level for all men who use the drug is less than the mean for those who do not use the drug. Assume both populations are normally distributed and use a 0.05 significance level.

Solution :

```
> no_drug<-c(237,289,257,228,303,275,262,304,244,233)
> drug <-c(194,240, 230, 166, 265, 222, 242, 281,240,212)
> t.test(no_drug, drug, alternative="greater",mu=0)
     Welch Two Sample t-test
data:  no_drug and drug
t = 2.4564, df = 17.587, p-value = 0.01234
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 9.967098      Inf
sample estimates:
mean of x mean of y
    263.2     229.2
```

2 Proportions - using x's and n's:

a) Hypothesis Tests for Two Proportions (prop.test):

Evaluates if the difference between two sample proportions is statistically significant, often applied in studies comparing proportions of categorical outcomes in two independent groups.

Example : A popular cold-remedy was tested for it's ecacy. In a sample of 150 people who took the remedy upon getting a cold, 117 (78%) had no symptoms one week later. In a sample of 125 people who took the placebo upon getting a cold, 90 (75%) had no symptoms one week later. The table summarizes this information.

group	# who are Symptom Free after one week (x)	total # in group (n)	proportion $\hat{p} = x/n$
Remedy	117	150	0.78
Placebo	90	120	0.75

The Test: Test the claim that the proportion of all remedy users who are symptom-free after one week is greater than the proportion for placebo users. Test this claim at the 0.05 significance level.

Solution :

```
> x<-c(117,90)
> n<-c(150,120)
> prop.test(x, n,alternative="greater",correct=FALSE)
     2-sample test for equality of proportions without continuity
     correction
```

```
data:  x out of n
X-squared = 0.3354, df = 1, p-value = 0.2812
alternative hypothesis: greater
95 percent confidence interval:
 -0.05557192  1.00000000
sample estimates:
prop 1 prop 2
  0.78   0.75
```

3 Using Test Statistics:

a) Tests about Two Means (pt and qt):

Utilizes the t-distribution to assess hypotheses about the difference between two population means from independent samples, with unknown population standard deviations.

Example :

```
> A <- c(65, 68, 70, 72, 68)
> B <- c(60, 62, 64, 66, 63)
>
t<-(mean(A)-mean(B))/sqrt((var(A)/length(A))+(var(B)/length(B)))#t-scor
e
> t
[1] 3.64529
> df <- length(A) + length(B) - 2# Degrees of freedom
> df
[1] 8
> 2 * pt(abs(t), df = df, lower.tail = FALSE)# two-tailed p-value
[1] 0.006538744
```

b) Tests about Two Proportions (pnorm and qnorm):

Employs the normal distribution to evaluate hypotheses about the difference between two population proportions, often applied in large sample sizes for proportions from independent groups.

Example :

```
> n_A <- 100  # Sample size for group A
> n_B <- 150  # Sample size for group B
> success_A <- 55  # No of successes in A
> success_B <- 70  # No of successes in B
> p_A <- success_A / n_A#Proportions
> p_B <- success_B / n_B
> z_<-(p_A-p_B)/sqrt((p_A*(1-p_A))/n_A+(p_B*(1-p_B))/n_B)
> z
[1] -1.264911
> 2 * pnorm(abs(z),lower.tail = FALSE)# two-tailed p-value
[1] 0.2059032
```

Conclusion - Studied and Implemented the Hypothesis Testing (1 Sample & 2 Sample) in R programming language Successfully.