lendi Institute of Engineering & Technology
An Autonomous Institution
Accredited by NAAC with " A" Grade, Accredited by NBA (ECE, CSE.EEE & MECH)
Approved by A.I.C.T.E. & Permanently Affiliated to J. N. T. U. Gurajada, VIZIANAGARAM
Via 5th APSP Battalion, Jonnada (V), Denkada (M), NH-3, Vizianagaram Dist - 535005, A.P. Website : www.lendi.org
Ph : 08922-241111, 241666, Cell No : 9490344747, 9490304747,e-mail : lendi_2008@yahoo.com

R-PROGRAMMING

LAB MANUAL

DEPARTMENTOFCOMPUTERSCIENCEAND

INFORMATION TECHNOLOGY

lendi Institute of Engineering & Technology
An Autonomous Institution
Accredited by NAAC with " A" Grade, Accredited by NBA (ECE, CSE.EEE & MECH)
Approved by A.I.C.T.E. & Permanently Affiliated to J. N. T. U. Gurajada, VIZIANAGARAM
Via 5th APSP Battalion, Jonnada (V), Denkada (M), NH-3, Vizianagaram Dist - 535005, A.P. Website : www.lendi.org
Ph : 08922-241111, 241666, Cell No : 9490344747, 9490304747,e-mail : lendi_2008@yahoo.com

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLGOY

VISION

To excel in computing arena and to produce globally competent computer science and Information Technology graduates with Ethical and Human values to serve the society

MISSION

- To impart strong theoretical and practical background in computer science and information technology discipline with an emphasis on software development.
- To provide an open environment to the students and faculty that promotes professional growth
- To inculcate the skills necessary to continue their education and research for Contribution to society.

lendi Institute of Engineering & Technology
An Autonomous Institution
Accredited by NAAC with " A" Grade, Accredited by NBA (ECE, CSE.EEE & MECH)
Approved by A.I.C.T.E. & Permanently Affiliated to J. N. T. U. Gurajada, VIZIANAGARAM
Via 5th APSP Battalion, Jonnada (V), Denkada (M), NH-3, Vizianagaram Dist - 535005, A.P. Website : www.lendi.org
Ph : 08922-241111, 241666, Cell No : 9490344747, 9490304747,e-mail : lendi_2008@yahoo.com

COURSE OUTCOMES (CO's)

CO1: Apply the concepts of data types, data structure and advanced data structure in R-Programming to the basic mathematics.

CO2: Implement R programming codes using control statements.

CO3: Implement R programming codes using functions.

CO4: Implement R programming code to Mathematical and Statistical techniques.

CO5: Create various graphs using R Programming codes for data interpretations.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: Graduates of Computer Science and Information Technology will acquire strong knowledge to analyze, design, and develop computing products and solutions for real-life problems utilizing the latest tools, techniques, and technologies.

PEO2: Graduates of Computer Science and Information Technology shall have interdisciplinary approach, professional attitude and ethics, communication, teamwork skills and leadership capabilities to solve social issues through their Employment, Higher Studies and Research.

PEO3: Graduates will engage in life-long learning and professional development to adapt to dynamically computing environment.

lendi Institute of Engineering & Technology
An Autonomous Institution
Accredited by NAAC with " A" Grade, Accredited by NBA (ECE, CSE.EEE & MECH)
Approved by A.I.C.T.E. & Permanently Affiliated to J. N. T. U. Gurajada, VIZIANAGARAM
Via 5th APSP Battalion, Jonnada (V), Denkada (M), NH-3, Vizianagaram Dist - 535005, A.P. Website : www.lendi.org
Ph : 08922-241111, 241666, Cell No : 9490344747, 9490304747,e-mail : lendi_2008@yahoo.com

PROGRAM OUTCOMES (PO's)

PO1: Engineering Knowledge	PO7:Environment & Sustainability
PO2: Problem Analysis	PO8: Ethics
PO3: Design & Development	PO9: Individual & Team Work
PO4: Investigations	PO10: Communication Skills
PO5: Modern Tools	PO11: Project Mgt & Finance
PO6:Engineer & Society	PO12: Life Long Learning

PROGRAM SPECIFIC OUTCOMES (PSO's)

PSO1: Ability to solve contemporary issues utilizing skills.
PSO2:To acquire knowledge of the latest tools and technologies to provide technical solutions
PSO3:Toqualify in national and international competitive examinations for successful higher studies and employment.

LAB SYLLABUS

Introduction

How to run R, R Sessions and Functions, Basic Math, Variables,
Data Types: Vectors and Conclusion, Advanced Data Structures: Data Frames, Lists and Matrices.
1. Write a program to illustrate basic Arithmetic in R
2. Write a program to illustrate operations on a vector in R
3. Write a program to illustrate operations on matrix in R
5. Write a program to illustrate operations on Data Frame
6. Write a program to illustrate operations on List

R Programming Structures

R Programming Structures, Control Statements, Loops, If-Else, and Functions.
1. Write a program to illustrate if-else-else if in R
2. Write a Program to illustrate While and For loops in R
3. Write a program to illustrate Functions in Quick sort implementation in R
4. Write a program to illustrate Function inside function in R

Doing Math and Simulation in R

R Programming implementation for basic Mathematics, Probability and Statistical methods, Correlation, Regression and Statistical Distributions.
1. Write a program to illustrate built-in function for mathematics in R.
2. Write a program to illustrate built-in function for probabilities in R.
3. Write a program to illustrate built-in function for correlation and regression lines in R.
4. Write a program to illustrate built-in function for statistical distributions in R.

Graphics

Graphics, Creating Graphs, The Workhorse of R Base Graphics, the plot () Function, Customizing Graphs, Saving Graphs to Files. Import the data sets from excel, CSV to visualize the data and applying statistical methods.
1. Write a program to illustrate to create graphs and usage of plot() function in R
2. Write a program to illustrate Customizing and Saving to Graphs in R.
3. Write a program to illustrate the data sets by applying statistical methods.
4. Write a program to illustrate the data sets by data visualization

COURSE OUTCOMES Vs PO's & PSO's:

SNO	DESCRIPTION	PO(1..12) MAPPING	PSO(1,2) MAPPING
CO1	Apply the concepts of data types, data structure and advanced data structure in R-Programming to the basic mathematics.	PO1, PO2, PO3, PO4, PO5, PO9, PO12	,PSO1, PSO2, PSO3
CO2	Implement R programming codes using control statements.	PO1, PO2, PO3, PO4, PO5, PO9, PO12	PSO1, PSO2, PSO3
CO3	Implement R programming codes using functions.	PO1, PO2, PO3, PO4, PO5, PO9, PO12	PSO1, PSO2, PSO3
CO4	Implement R programming code to Mathematical and Statistical techniques.	PO1, PO2, PO3, PO4, PO5, PO9, PO12	PSO1, PSO2, PSO3
CO5	Create various graphs using R Programming codes for data interpretations.	PO1, PO2, PO3, PO4, PO5, PO9, PO12	PSO1, PSO2, PSO3
COURSE OVERALL PO/PSO MAPPING: PO1, PO2, PO3, PO4, PO5, PO9, PO12, PSO1, PSO2 and PSO3			

COURSE OUTCOMES VS POs MAPPING (DETAILED; HIGH:3; MEDIUM:2; LOW:1**):**

SNO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	3	3	2	1				1			1	2	2	2
CO2	3	3	3	2	1				1			1	2	2	2
CO3	3	3	3	2	1				1			1	2	2	2
CO4	3	3	3	2	1				1			1	2	2	2
CO5	3	3	3	2	1				1			1	2	2	2
CO*	3	3	3	2	1				1			1	2	2	2

lendi Institute of Engineering & Technology
An Autonomous Institution
Accredited by NAAC with " A" Grade, Accredited by NBA (ECE, CSE.EEE & MECH)
Approved by A.I.C.T.E. & Permanently Affiliated to J. N. T. U. Gurajada, VIZIANAGARAM
Via 5th APSP Battalion, Jonnada (V), Denkada (M), NH-3, Vizianagaram Dist - 535005, A.P. Website : www.lendi.org
Ph : 08922-241111, 241666, Cell No : 9490344747, 9490304747,e-mail : lendi_2008@yahoo.com

SYLLABUS INDEX

SNO	PROGRAM DESCRIPTION	Page No
1	1.1 Write a program to illustrate basic Arithmetic in R. 1.2 Write a program to illustrate operations on a vector in R. 1.3 Write a program to illustrate operations on matrix in R. 1.4 Write a program to illustrate operations on Data Frame. 1.5 Write a program to illustrate operations on List.	9 - 28
2	2.1 Write a program to illustrate if-else-else if in R 2.2 Write a Program to illustrate While and For loops in R 2.3 Write a program to illustrate Functions in Quick sort implementation in R. 2.4 Write a program to illustrate Function inside function in R.	29 - 41
3	3.1 Write a program to illustrate built-in function for mathematics in R. 3.2 Write a program to illustrate built-in function for probabilities in R. 3.3 Write a program to illustrate built-in function for correlation and regression lines in R. 3.4 Write a program to illustrate built-in function for statistical distributions in R.	42 - 57
4	4.1 Write a program to illustrate to create graphs and usage of plot() function in R. 4.2 Write a program to illustrate Customizing and Saving to Graphs in R. 43 Write a program to illustrate the data sets by applying statistical methods. 4.4 Write a program to illustrate the data sets by data visualization.	58 - 70

Instructions to students:

Pre-lab activities:

- Prepare observation note book which contains the following :
- Procedure/algorithm/program to solve the problems discussed in the theory class.
- Solutions to the exercises given in the previous lab session.
- Refer the topics covered in theory class.

In-lab activities:

- Note down errors observed while executing program and remedy for that.
- Note down corrections made to the code during the lab session.
- Answer to vivo-voce.
- Get the observation corrected.
- Note down inferences on the topic covered by the programs executed.

Post-lab activities:

- Solve the given exercises.
- Devise possible enhancements that can be made to the solved problem to simplify the logic
- Executed programs should be recorded in the lab record and corrected within one week after completion of the experiment.
- After completion of every module, attest will be conducted, and assessment results will have weight in the final internal marks.

General Instructions:

- Student should sign in the log register before accessing the system.
- Student is only responsible for any damage caused to the equipment in the laboratory during his session.
- Usage of pen drives is not allowed in the lab.
- If a problem is observed in any hardware equipment, please report to the lab staff immediately; do no attempt to fix the problem yourself.
- Systems must be shut down properly before leaving the lab.
- Please be considerate of those around you, especially in terms of noise level. While labs area natural place for conversations regarding programming, kindly keep the volume turned off.

Module: 1

Introduction, How to Run R, R Sessions and Functions, Basic Math, Variables, Data Types: Vectors and Conclusion, Advanced Data Structures: Data Frames, Lists and Matrices.

1. Write a program to illustrate basic Arithmetic in R?
2. Write a program to illustrate operations on a vector in R?
3. Write a program to illustrate operations on matrix in R?
4. Write a program to illustrate operations on Data Frame in R?
5. Write a program to illustrate operations on Lists in R?

Aim: Write a program to illustrate basic Arithmetic in R?

Description: The R Arithmetic operations includes operators like Arithmetic addition, subtraction,
multiplication and division. All these R arithmetic operators are binary operators. which means they operate on two operands. In this program we are using two variables a and b to perform arithmetic operations present in R.

Arithmetic Operators:

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division

Relational Operators:

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to

Algorithm:

```
1. Start
2. Perform Arithmetic Operations:
     o Initialize a <- 10
     o Initialize b <- 8
     o Compute and store results:
          ▪ sum <- a + b
          ▪ sub <- a - b
          ▪ multi <- a * b
          ▪ div <- a / b
     o Print the results:
          ▪ print(sum)
          ▪ print(sub)
          ▪ print(multi)
          ▪ print(div)
3. Perform Relational Operations:
     o Case 1: Check Inequality (!=):
          ▪ Assign var1 <- 5
          ▪ Assign var2 <- 7
          ▪ Print the result of var1 != var2
     o Case 2: Check Greater Than (>):
          ▪ Assign var1 <- 5
          ▪ Assign var2 <- 7
          ▪ Print the result of var1 > var2
     o Case 3: Check Less Than (<):
          ▪ Assign var1 <- 5
          ▪ Assign var2 <- 7
          ▪ Print the result of var1 < var2
     o Case 4: Check Greater Than or Equal To (>=):
          ▪ Assign var1 <- 5
          ▪ Assign var2 <- 7
          ▪ Print the result of var1 >= var2
     o Case 5: Check Less Than or Equal To (<=):
          ▪ Assign var1 <- 5
          ▪ Assign var2 <- 7
          ▪ Print the result of var1 <= var2
     o Case 6: Check Equality (==):
          ▪ Assign var1 <- 10
          ▪ Assign var2 <- 10
          ▪ Print the result of var1 == var2
4. End
```

Aim: Write a program to illustrate operations on a vector in R?

Description: Vectors are the most basic data types in R even a single object created is also stored in the form of a vector.
Vectors are nothing but arrays as defined in other languages vectors contain a sequence of Homogeneous types of data.

Basic R vectors are objects are 6 types:

1. Character eg:"ABC"

2. Double eg: 12.9

3.Complex Eg: 2+8i

4.Raw

Eg: charToRaw("val")5.Integer

eg: 45L

5. Logical eg: TRUE/FALSE

Algorithm:

1. **Start**
2. **Create and Print Vectors of Different Types:**
   - **Character Vector:**
     - Print the string "abc":
       - print("abc")
   - **Double Vector:**
     - Print the double value 12.5:
       - print(12.5)
   - **Integer Vector:**
     - Print the integer value 63L:
       - print(63L)
   - **Logical Vector:**
     - Print the logical value TRUE:
       - print(TRUE)
   - **Complex Vector:**
     - Print the complex value 2 + 3i:
       - print(2 + 3i)
   - **Raw Vector:**
     - Convert the string "hello" to its raw representation and print:
       - print(charToRaw("hello"))

3. **Perform Arithmetic Operations on Vectors:**
    - **Vector Addition:**
        - Define two vectors:
            - v1 <- c(3, 8, 4, 5, 0, 11)
            - v2 <- c(4, 11, 0, 8, 1, 2)
        - Compute their element-wise sum:
            - add.result <- v1 + v2
        - Print the result:
            - print(add.result)
    - **Vector Subtraction:**
        - Compute their element-wise difference:
            - sub.result <- v1 - v2
        - Print the result:
            - print(sub.result)
    - **Vector Multiplication:**
        - Compute their element-wise product:
            - multi.result <- v1 * v2
        - Print the result:
            - print(multi.result)
    - **Vector Division:**
        - Compute their element-wise division:
            - div.result <- v1 / v2
        - Print the result:
            - print(div.result)
4. **End**

Aim: Write a program to illustrate operations on matrix in R?

Description: Matrices in R are a bunch of values, either real or complex numbers, arranged in a group of fixed number of rows and number of columns. Matrices are used to depict the data in toa structured and well-organized format. It is necessary to enclose the elements of a matrix in paranthesis or brackets.

Matrix can be created using matrix( ) function.

Eg:

2 4 6 9

4 5 6 3

6 7 8 5

Basic Syntax of matrix is: Matrix(data,nrow,ncol,byrow,dimnames)

Algorithm:

1. **Start**
2. **Create and Print a Matrix:**
    - Define a matrix P with values from 3 to 14, arranged in 4 rows, filled row-wise:
        - P <- matrix(c(3:14), nrow = 4, byrow = TRUE)
    - Print the matrix:
        - print(P)
3. **Access Specific Elements in the Matrix:**
    - Access and print the element in the 3rd column and 1st row of P:
        - print(P[1, 3])
    - Access and print the element in the 2nd column and 4th row of P:
        - print(P[4, 2])
4. **Perform Arithmetic Operations Between Matrices:**
    - Create another matrix Q with values from 14 to 25, arranged in 4 rows, filled row-wise:
        - Q <- matrix(c(14:25), nrow = 4, byrow = TRUE)
    - **Matrix Addition:**
        - Compute the element-wise sum of P and Q:
            - print(P + Q)
    - **Matrix Subtraction:**
        - Compute the element-wise difference of P and Q:
            - print(P - Q)
    - **Matrix Multiplication:**
        - Compute the element-wise product of P and Q:
            - print(P * Q)
    - **Matrix Division:**
        - Compute the element-wise division of P by Q:
            - print(P / Q)
5. **End**

Aim: Write a program to illustrate operations on Data Frame in R?

Description: Data frames are generic data objects of R. which are used to store data table like structures that contains same as 2-dimensional matrix.

Matrix containing rows and columns each column contains values of each variable.

Each column contains values of each variable and each row contains one set of values related to each column.

The column names are non-empty and Row names should be unique.

Data frames are considered to be the most popular data objects in R.

Syntax: varname <-data.frame(values).

Algorithm:

1. **Start**
2. **Create a Data Frame:**
    - Define the data frame employee.data with the following columns:
        - employee_id: Values from 1 to 4.
        - employee_name: Names "raj", "rohit", "amit", "sahil".
        - salary: Values 63003, 51502, 61002, 44440.
    - Specify stringsAsFactors = FALSE to prevent conversion of strings to factors.
    - Print the data frame:
        - print(employee.data)
3. **Analyze the Data Frame:**
    - **Get the Structure of the Data Frame:**
        - Use str() to display the structure of employee.data:
            - str(employee.data)
    - **Get a Summary of the Data Frame:**
        - Use summary() to get statistical summaries of numerical columns and information about categorical columns:
            - summary(employee.data)
4. **Add a New Column to the Data Frame:**
    - Add a column department with values:
        - c("Network", "Software", "Mobile", "Web")
    - Assign the new column to employee.data:
        - employee.data$department <- c("Network", "Software", "Mobile", "Web")
    - Print the updated data frame:
        - print(employee.data)
5. **End**

Aim: Write a program to illustrate operations on Lists in R?

Description: Lists in R language, are the objects which comprises elements of divered types

like numbers, strings, logical values.

Lists with in a list and also matrix function as its elements.

The list plays a central role in R, forming thebasis for data frames, object-oriented programming, andso on. As with vectors and matrices, one common operation with lists is indexing.

Creating Lists:

(i) Creating a list with names of the components: Syntax: list(...)

(ii) Creating a list using vector():vector(mode="list")

There are three ways to access an individual component c of a list lst and return it in the datatype of c:

• lst$c
• lst[["c"]]
• lst[[i]], where i is the index of i within lst lst[i],
where i is the index of c within lst.

Algorithm:

1. **Start**
2. **Create a List:**
    - Create a list list_data containing various types of elements:
        - Strings: "Red", "Green"
        - A vector: c(21, 32, 11)
        - A logical value: TRUE
        - Numbers: 51.23, 119.1
    - Print the list:
        - print(list_data)
3. **Access Elements in the List:**
    - Access and print the first element of the list:
        - print(list_data[1])
    - Access and print the third element of the list:
        - print(list_data[3])
4. **Assign Names to List Elements:**
    - Assign names to the elements of the list:
        - names(list_data) <- c("1st Quarter", "A Matrix", "A Inner List")
5. **Add an Element to the List:**
    - Add a new element at the 4th position:
        - list_data[4] <- "New element"
    - Print the newly added element:
        - print(list_data[4])

6. **Remove an Element from the List:**
   - Remove the element at the 4th position by assigning NULL:
     - list_data[4] <- NULL
   - Attempt to print the removed element (should return NULL):
     - print(list_data[4])
7. **Update an Element in the List:**
   - Update the value of the third element to "updated element":
     - list_data[3] <- "updated element"
   - Print the updated third element:
     - print(list_data[3])
8. **End**

Viva Questions

1. What is R programming?
2. How do you install R and R Studio?
3. What are packages in R?
4. What is an R script?
5. What are the key features of R?
6. What is an R function?
7. What is a vector in R?
8. What is the difference between a matrix and a data frame in R?
9. What are Lists in R?
10. What are Data Frames in R?

Module - 2

R Programming Structures:

R-Programming Structures, Control Statements, Loops, If-Else, and Functions.

1. Write a program to illustrate if-else-else if in R.

2. Write a Program to illustrate While and For loops in R.

3. Write a program to illustrate Functions in Quick sort implementation in R.

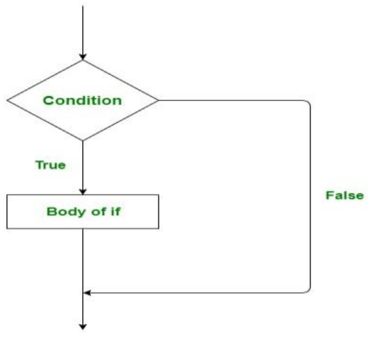
4. Write a program to illustrate Function inside function in R.

Aim: Write a Program to illustrate if-else and else-if in R?

Description: The if-statement in Programming Language alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won't. But what if we want to do something else if the condition is false? Here comes the R Programming Language **else statement**. We can use the else statement with the if statement to execute a block of code when the condition is false.

Syntax

```
If (condition) {
# code to be executed if condition is TRUE
} else {
# code to be executed if condition is FALSE
}
```



The else if statement is also known as nested if-else statement. The if statement is followed by an optional else if..... else statement. This statement is used to test various condition in a single if......else if statement. There are some key points which are necessary to keep in mind when we are using the if.....else if.....else statement.

1.if statement can have either zero or one else statement and it must come after any else if's statement.
2.if statement can have many else if's statement and they come before the else statement.
3.Once an else if statement succeeds, none of the remaining else if's or else's will be tested.

syntax :

```
1.if (boolean_expression 1) {
// This block executes when the boolean expression 1 is true.
2.} else if (boolean_expression 2) {
// This block executes when the boolean expression 2 is true.
3.} else if ( boolean_expression 3) {
// This block executes when the boolean expression 3 is true.
4.} else {
// This block executes when none of the above condition is true.
5.}
```

Algorithm:

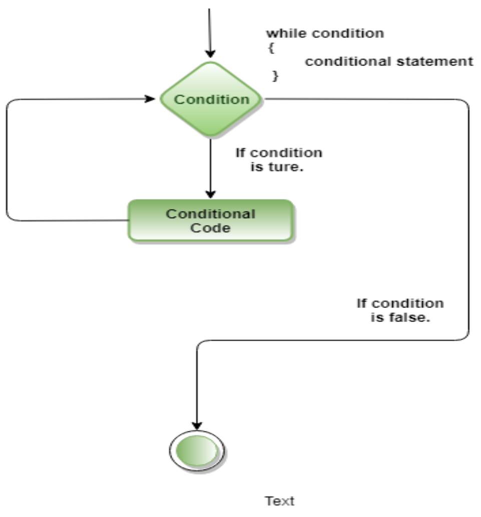
1. **Start**
2. **First If-Else Condition:**
   - Initialize the variable varx with the value 5.
   - Check if varx is equal to 5:
     - If true, print "True".
     - If false, print "False".
3. **Second If-Else Condition:**
   - Initialize the variable varx with the value 5.
   - Check if varx is not equal to 5:
     - If true, print "True".
     - If false, print "False".
4. **Comparison of Two Variables (Equal Condition):**
   - Initialize two variables x and y, both with the value 10.
   - Check if x is equal to y:
     - If true, print "x is equal to y".
     - If false, print "x is not equal to y".
5. **Comparison of Two Variables (Not Equal Condition):**
   - Initialize x with the value 10 and y with the value 1.
   - Check if x is equal to y:
     - If true, print "x is equal to y".
     - If false, print "x is not equal to y".
6. **Else-If Ladder:**
   - Initialize x with the value 5 and y with the value 10.
   - Check if x is equal to y:
     - If true, print "x is equal to y".
   - Else, check if x is greater than y:
     - If true, print "x is not equal to y, and x is greater than y".
   - Else, print "x is not equal to y, and x is less than y".
7. **End**

Aim: Write a program to illustrate while and For loops in R?

Description: While loop in R programming language is used when the exact number of iterations of a loop is not known beforehand. It executes the same code again and again until a stop condition is met. While loop checks for the condition to be true or false n+1 times rather than **n** times. This is because the while loop checks for the condition before entering the body of the loop.

Syntax:

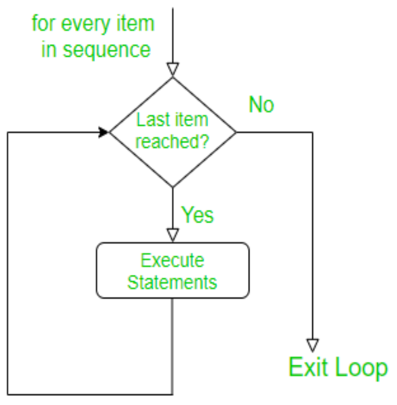
```
while (test_expression){
statement
update_expression
}
```



For loop in R Programming Language is useful to iterate over the elements of a list, data frame, vector, matrix, or any other object. It means the for loop can be used to execute a group of statements repeatedly depending upon the number of elements in the object. It is an entry-controlled loop, in this loop, the test condition is tested first, then the body of the loop is executed, the loop body would not be executed if the test condition is false.

Syntax:

```
for  (variable _ name)
{
   Statement(s)
}
```



For Loop in R

While Loop Algorithm:

1. **Start**
2. **First While Loop:**
    - Initialize a vector values with the element "while loop".
    - Initialize the variable count with the value 5.
    - While count is less than 7, repeat the following:
        - Print the vector values.
        - Increment count by 1 after each iteration.
    - Exit the loop once count reaches 7.
3. **Second While Loop:**
    - Initialize the variable i with the value 1.
    - While i is less than 6, repeat the following:
        - Print the value of i.
        - Increment i by 1 after each iteration.
    - Exit the loop once i reaches 6.
4. **End**

For Loop Algorithm:

1. **Start**
2. **First For Loop:**
    - Create a vector x containing the first 7 uppercase letters of the alphabet using LETTERS[1:7].
    -
    - For each element i in the vector x, repeat the following:
        - Print the value of i.
    - The loop will iterate through the letters from A to G (the first 7 letters).
3. **Second For Loop:**
    - For each integer i in the range from 0 to 10, repeat the following:
        - Print the value of i.
    - The loop will iterate from 0 to 10 and print each value.
4. **End**

Aim: Write a program to illustrate functions in Quick sort implementation in R?

Description: This algorithm follows the divide and conquer approach. Divide and conquer is a technique of breaking down the algorithms into sub problems, then solving the sub problems, and combining the results back together to solve the original problem.

Divide: In Divide, first pick a pivot element. After that, partition or rearrange the array into two sub-arrays such that each element in the left sub-array is less than or equal to the pivot element and each element in the right sub-array is larger than the pivot element.

Conquer: Recursively, sort two sub arrays with Quick sort.

Combine: Combine the already sorted array.

Quick sort picks an element as pivot, and then it partitions the given array around the picked pivot element. In quick sort, a large array is divided into two arrays in which one holds values that are smaller than the specified value (Pivot), and another array holds the values that are greater than the pivot.

After that, left and right sub-arrays are also partitioned using the same approach. It will continue until the single element remains in the sub-array.

Algorithm:

1. **Start**
2. **Define the Quick Sort Function:**
    - Define the function quicksort(arr) that takes an array arr as input:
        - If the length of arr is less than or equal to 1, return the array as it is already sorted.
        - Otherwise:
            1. **Choose a Pivot Element:**
                - Select the first element of the array arr as the pivot.
            2. **Partition the Array:**
                - Create three subarrays:
                    - smaller: All elements less than the pivot.
                    - equal: All elements equal to the pivot.
                    - larger: All elements greater than the pivot.
            3. **Recursive Sorting:**
                - Recursively apply the quicksort function to the smaller and larger subarrays:
                    - sorted_smaller <- quicksort(smaller)
                    - sorted_larger <- quicksort(larger)
                    -
            4. **Combine the Sorted Sub arrays:**
                - Concatenate the sorted smaller, equal, and larger subarrays:
                    - return(c(sorted_smaller, equal, sorted_larger))
3. **Apply the Quick Sort Function:**
    - Define the input array elements with the values c(7, 6, 7, 5, 9, 2, 1, 15, 10).
    - Call the quicksort function to sort the elements array:

        - sorted_elements <- quicksort(elements)
4. **Print the Sorted Elements:**
    - Print the sorted array sorted_elements:
        - print(sorted_elements)
5. **End**

Aim: Write a program to illustrate function inside function in R.

Description: In R Programming , a **function inside a function** refers to a scenario where a function is defined and used within the body of another function. This is also known as a **nested function**.
1.Scope: The inner function is local to the outer function and cannot be accessed outside of it.

2.Purpose: It is often used to encapsulate logic that is specific to the outer function, improving modularity and readability.

Algorithm:

1. **Start**
2. **Define the First Function (afun):**
   - Define the function afun(a) that takes one parameter a:
     - Use a for loop to iterate from 1 to a:
       1. Multiply i by 2 and assign it to b.
       2. Print the value of b.
3. **Call the afun Function:**
   - Call afun(3) to execute the function with the argument 3.
   - The function will print 2, 4, and 6 (since 1*2 = 2, 2*2 = 4, 3*2 = 6).
4. **Define the Second Function (calculate_tax):**
   - Define the function calculate_tax(tax_rate) that takes one parameter tax_rate:
     - Define an inner function apply_tax(income) within calculate_tax:
       - **Inner Function**:
         - Calculate the tax amount by multiplying income by tax_rate.
         - Subtract the tax amount from the income to get the taxed_income.
         - Return the taxed_income.

Return the inner function apply_tax as the result of calculate_tax.

5.Create a Tax Calculator:

Call calculate_tax(0.20) with a 20% tax rate:
Store the returned function in the variable tax_calculator.

6.Use the Tax Calculator:

Define the variable income with the value 1000.
Call tax_calculator(income) to calculate the taxed income:
Store the result in taxed_income.

7.Display the Results:

Print the original income and the taxed income using cat():
cat("Original Income:", income, "\n")
cat("Taxed Income:", taxed_income, "\n")

8.End

Viva Questions

1. What are R Programming Structures?
2. What are Control Statements in R?
3. What are Loops in R?
4. How do you define functions in R?
5. What is the if-else **statement in R?**
6. What is a matrix in R?
7. What are the main data structures in R?
8. What is the purpose of the str() **function in R?**
9. What is a factor in R?
10. How do you bind vectors, rows, or columns in R?

MODULE - 3

Doing Math and Simulation in R:

R Programming implementation for basic Mathematics, Probability and Statistical methods, Correlation, Regression and Statistical Distributions.

1. Write a program to illustrate built-in function for mathematics in R.

2. Write a program to illustrate built-in function for probabilities in R.

3. Write a program to illustrate built-in function for correlation and regression lines in R.

4. Write a program to illustrate built-in function for statistical distributions in R.

Aim: Write a program to illustrate built-in function for mathematics in R.

Description:
1.min ( ) and max ( ) : It returns the smallest value in a vector or collection of values.

2.sum( ) : The sum() function in R calculates the sum of all the values in a vector.

3.mean( ) : It computes the average (mean) of a vector, i.e., It adds up all the values present in the vector and divides the total by the number of values.

4.sqrt( ): It computes the square root of a number or a vector of numbers.

5.abs( ) : It returns the absolute value of a number or a vector of numbers.

6.ceiling( ) : The ceiling ( ) function rounds up a number to the nearest integer greater than or equal.

7.floor ( ) : The floor ( ) function rounds down a number to the nearest integer less than or equal.

8.trunc ( ): It removes all decimal places from a number and reduces it to its integer portion.

9.round ( ) : The round( ) function rounds an integer to the number of decimal places provided.

10.cos( ),sin( ),tan( ): The cos( ), sin( ), and tan( ) functions compute the cosine, sine, and tangent of an angle in radians, respectively.

11.log( ) and log10( ) : It computes the natural logarithm of a number or a vector of numbers.

12.exp ( ): The exp( ) function calculates the exponential value (raised to the power of x) for a number or a vector of numbers.

Algorithm:

1. **Start**
2. **Find Minimum and Maximum Values:**
    - Define a vector values with the elements c(3, 12, 15, 1, 18, 21).
    - Use the min( ) function to calculate the minimum value in values.
    - Print the minimum value.
    - Use the max( ) function to calculate the maximum value in values.
    - Print the maximum value.
3. **Calculate Sum of Values:**
    - Define the vector values again with the elements c(3, 12, 15, 1, 18, 21).
    - Use the sum( ) function to calculate the total sum of values.
    - Print the total sum.
4. **Compute the Average (Mean):**
    - Define the vector values again with the elements c(3, 12, 15, 1, 18, 21).
    - Use the mean( ) function to calculate the average of values.
    - Print the average value.
5. **Calculate Square Root:**
    - Define a variable x with the value 36.
    - Use the sqrt( ) function to compute the square root of x.
    - Print the square root.
6. **Get the Absolute Value:**
    - Define a variable x with the value -15.
    - Use the abs( ) function to compute the absolute value of x.
    - Print the absolute value.
7. **Round Up to the Nearest Integer:**
    - Define a variable x with the value 8.7.
    - Use the ceiling( ) function to round x up to the nearest integer.
    - Print the rounded value.
8. **Round Down to the Nearest Integer:**
    - Define a variable x with the value 5.8.
    - Use the floor( ) function to round x down to the nearest integer.
    - Print the rounded value.
9. **Remove Decimal Places:**
    - Define a variable x with the value 4.7.
    - Use the trunc( ) function to remove the decimal part of x.
    - Print the truncated value.
10. **Round to a Specified Number of Decimal Places:**
    - Define a variable x with the value 3.14266.
    - Use the round( ) function to round x to 2 decimal places.
    - Print the rounded value.
11. **Calculate Trigonometric Functions:**
    - Define an angle variable with the value 45.
    - Use the cos( ) function to calculate the cosine of angle.
    - Use the sin( ) function to calculate the sine of angle.
    - Use the tan( ) function to calculate the tangent of angle.
    - Print the cosine, sine, and tangent values.

12. **Compute Logarithms:**
    - Use the log( ) function to calculate the natural logarithm of 10.
    - Print the result of the natural logarithm.
    - Use the log10( ) function to calculate the base-10 logarithm of 100.
    - Print the result of the base-10 logarithm.
13. **Calculate Exponential Value:**
    - Define a variable x with the value 2.
    - Use the exp( ) function to calculate the exponential value of x (e raised to the power of x).
    - Print the exponential value.
14. **End**

Aim: Write a program to illustrate built-in function for probabilities in R.

Description: R provides extensive built-in probabilities functions, allowing programmers to analyze and work with probability distributions. These functions include **normal, binomial, Poisson**, and **uniform distribution**.
We generate a random sample of 100 values from a normal distribution using **rnorm**.
We calculate the cumulative distribution function (CDF) and probability density function (PDF) for a given value using **pnorm** and **dnorm**, respectively.
We generate random numbers from a discrete distribution using **sample**.
We generate random numbers from a discrete distribution.
We simulate data from a binomial distribution using **rbinom**.

Algorithm:

1. **Start**
2. **Set Seed for Reproducibility:**
   - Use set.seed(123) to ensure the results are reproducible in subsequent runs of the program.
3. **Generate Random Sample from Normal Distribution:**
   - Generate a random sample of 100 values from a normal distribution using the rnorm( ) function.
   - Set the mean to 0 and the standard deviation to 1 for the normal distribution.
   - Store the generated values in the variable data.
4. **Calculate Cumulative Distribution Function (CDF):**
   - Define a variable x with the value 1.
   - Use the pnorm( ) function to calculate the cumulative distribution function (CDF) for the value x based on the mean and standard deviation of the data.
   - Print the result in the format P(X <= x).
5. **Calculate Probability Density Function (PDF):**
   - Use the dnorm( ) function to calculate the probability density function (PDF) for the value x based on the mean and standard deviation of the data.
   - Print the result in the format f(X = x).
6. **Generate Random Numbers from a Discrete Distribution:**
   - Define a vector categories with the values c("A", "B", "C").
   - Define a vector probabilities with the corresponding probabilities c(0.3, 0.5, 0.2).
   - Use the sample( ) function to randomly select one category from categories based on the provided probabilities.
   - Store the selected category in the variable random_category and print the result.
7. **Simulate a Binomial Distribution:**
   - Define variables num_trials with the value 10 and prob_success with the value 0.3.
   - Use the rbinom( ) function to simulate data from a binomial distribution with num_trials trials and a probability of success prob_success.
   - Store the simulated result in the variable simulated_data and print the result.
8. **End**

Aim: Write a program to illustrate built-in function for correlation and regression lines in R.

Description: In R programming, correlation and regression lines are used often to analyze relationships between variables.

1. Correlation Line:

The ‘cor( )’ function in R is used to calculate the correlation coefficient between two variables.

Syntax: x <- (1,2,3,4,5)

Y <-(2,4,5,4,5)

Correlation _ coefficient <- cor (x,y)

2. Regression Line: The ‘lm ( )’ function is commonly used for linear regression. The ‘abline ( )’ function is used to add the regression line to the scatter plot. Regression _ model <- lm (y ~ x)

Syntax:

Summary (regression _model)

Plot(x,y)

Abline (regression _model,col=“red”)

Algorithm:

1. **Start**
2. **Define Sample Data:**
   - Create two vectors x and y:
     - x <- c(1, 2, 3, 4, 5)
     - y <- c(2, 4, 5, 4, 5)
3. **Calculate the Correlation Coefficient:**
   - Use the cor( ) function to calculate the Pearson correlation coefficient between x and y.
   - Store the result in the variable correlation_coefficient
4. **Fit a Linear Regression Model:**
   - Use the lm( ) function to fit a linear regression model with y as the dependent variable and x as the independent variable.
   - Store the regression model in the variable regression_model.
5. **Print Regression Summary:**
   - Use the summary( ) function to display the details of the regression model, such as coefficients, residuals, and R-squared value.

6. **Create a Scatter Plot:**
   - Use the plot( ) function to create a scatter plot of x versus y with the title "Scatterplot with Regression Line".
7. **Add the Regression Line to the Plot:**
   - Use the abline( ) function to add the regression line to the scatter plot.
   - Set the color of the regression line to red (col = "red").
8. **End**

Aim: Write a program to illustrate built-in function for statistical distributions in R.

Description: R provides extensive statistical probability functions, allowing programmers to analyze and work with probability distributions. These functions include **normal, binomial, Poisson**, and **uniform distribution**. We can calculate cumulative probabilities, quantiles, and densities and generate random numbers using these functions.

Let us see the examples given below.

1. **pnorm( ) :** It calculates a given number's cumulative probability(areaunder the curve) in a standard normal distribution.

2. **qnorm( ):** It calculates a given probability's quantile (inverse cumulative probability) in a standard normal distribution.
3. **dnorm( ):** It calculates a given number's density(probability mass)in a standard normal distribution.
4. **rnorm( ):** It generates random numbers from a standard normal distribution.
5. **dbinom( ):** It calculates the binomial distribution's probability density function (PDF).

6. **pbinom( ):** It determines the cumulative probability of an event.

7. **qbinom( ):** It finds a particular number from the binomial distribution corresponding to a given cumulative probability value, p.
8. **rbinom( ):** It generates nrandom values from a binomial distribution using trials and probability of success on each trial (prob).
9. dpois( ): It calculates the probability of obtaining a specific number of successes x in a given period, where the parameter lambda (λ) represents the expected number of events.
10. **ppois( ):** It calculates the cumulative probability of observing less than or equal to q successes in a given period, where the parameter lambda (λ) represents the expected number of events.
11. **rpois( ):** It generates n random numbers from a Poisson distribution.
12. dunif( ): It provides information about the uniform distribution on the interval from min to max.

13. **punif( ):** It gives the cumulative distribution function(CDF) of the uniform distribution on the interval (min to max).
14. **qunif( ):** It gives the quantity function of the uniform distribution on the interval(min to max).
15. **runif( ):** It produces random numbers from a uniform distribution on the interval(min to max).

Algorithm:

1. **Start**
2. **pnorm( ) - Cumulative Distribution Function (CDF) of Normal Distribution:**
   - Define x as 4.78.
   - Use the pnorm() function to calculate the cumulative probability for x.
   - Store the result in cum_prob and print it.

3. **qnorm( ) - Quantile Function (Inverse CDF) of Normal Distribution:**
   - Define x as 0.75.
   - Use the qnorm() function to find the quantile corresponding to x.
   - Store the result in quant and print it.
4. **dnorm( ) - Probability Density Function (PDF) of Normal Distribution:**
   - Define x as 1.43.
   - Use the dnorm() function to calculate the probability density for x.
   - Store the result in dens and print it.
5. **rnorm( ) - Generate Random Numbers from Normal Distribution:**
   - Define n as 10.
   - Use the rnorm() function to generate 10 random numbers from a standard normal distribution.
   - Store the result in rnum and print it.
6. **dbinom( ) - Probability Mass Function (PMF) of Binomial Distribution:**
   - Define x as 3, size as 10, and prob as 0.3.
   - Use the dbinom() function to calculate the probability of getting exactly x successes in size trials with probability prob.
   - Store the result in dens and print it.
7. **pbinom( ) - Cumulative Distribution Function (CDF) of Binomial Distribution:**
   - Define q as 20, size as 35, and prob as 0.5.
   - Use the pbinom() function to calculate the cumulative probability of q successes in size trials with probability prob.
   - Store the result in cump and print it.
8. **qbinom( ) - Quantile Function (Inverse CDF) of Binomial Distribution:**
   - Define p as 0.35, size as 30, and prob as 0.45.
   - Use the qbinom() function to find the quantile corresponding to probability p in a binomial distribution.
   - Store the result in num and print it.
9. **rbinom( ) - Generate Random Numbers from Binomial Distribution:**
   - Define n as 5, size as 50, and prob as 0.5.
   - Use the rbinom() function to generate n random numbers from a binomial distribution with size trials and prob success probability.
   - Store the result in num and print it.
10. **dpois( ) - Probability Mass Function (PMF) of Poisson Distribution:**
    - Define x as 3 and lambda as 3.
    - Use the dpois() function to calculate the probability mass for x events in a Poisson distribution with rate lambda.
    - Store the result in prob and print it.
11. **ppois( ) - Cumulative Distribution Function (CDF) of Poisson Distribution:**
    - Define q as 4 and lambda as 4.
    - Use the ppois() function to calculate the cumulative probability of getting q or fewer events in a Poisson distribution with rate lambda.
    - Store the result in prob and print it.
12. **rpois( ) - Generate Random Numbers from Poisson Distribution:**
    - Define n as 5 and lambda as 5.
    - Use the rpois() function to generate n random numbers from a Poisson distribution with rate lambda.
    - Store the result in rnum and print it.

13. **dunif( ) - Probability Density Function (PDF) of Uniform Distribution:**
    - Define x as 5, min as 0, and max as 1.
    - Use the dunif() function to calculate the probability density at x for a uniform distribution with min and max as bounds.
14. **punif( ) - Cumulative Distribution Function (CDF) of Uniform Distribution:**
    - Define q as 0.7, min as 0, and max as 1.
    - Use the punif() function to calculate the cumulative probability of q for a uniform distribution with min and max bounds.
    - Store the result in prob and print it.
15. **qunif( ) - Quantile Function (Inverse CDF) of Uniform Distribution:**
    - Define p as 0.3, min as 0, and max as 1.
    - Use the qunif() function to find the quantile corresponding to probability p in a uniform distribution.
    - Store the result in val and print it.
16. **runif( ) - Generate Random Numbers from Uniform Distribution:**
    - Define x as 5, min as 0, and max as 1.
    - Use the runif() function to generate x random numbers from a uniform distribution between min and max.
    - Store the result in rnum and print it.
17. **End**

Viva Questions

1. How is Basic Mathematics implemented in R?
2. How is Correlation calculated in R?
3. How do you perform Regression in R?
4. What are Statistical Distributions in R?
5. What is the role of simulation in R?
6. How can you perform matrix operations in R?
7. How do you generate a random sample from a probability distribution in R?
8. What are some commonly used mathematical functions in R?

MODULE - 4

Graphics

Graphics, Creating Graphs, The Workhorse of R Base Graphics, the plot ( )

Function, Customizing Graphs, Saving Graphs to Files. Import the data sets from excel,

CSV to visualize the data and applying statistical methods.

1. Write a program to illustrate to create graphs and usage of plot( ) function in R
2. Write a program to illustrate Customizing and Saving to Graphs in R.
3. Write a program to illustrate the data sets by applying statistical methods.
4. Write a program to illustrate the data sets by data visualization.

Aim: Write a program to illustrate to create graphs and usage of plot( ) function in R

Description: The most commonly used graphs in the R Programming language are scattered

Plots, box plots and charts, pie charts and bar charts, R Programming graphs supports both

Two dimensional and three dimensional plots.

The plot( ) function is used to draw points (markers) in a diagram.

The function takes parameters for specifying points in the diagram.

Parameter 1 specifies points on the x-axis**.**

Parameter 2 specifies points on the y-axis**.**

Algorithm:

1. **Start**
2. **Create a Simple Graph with Plot()**
   - Use the plot() function with two arguments:
     - The x-coordinate is 2.5 and the y-coordinate is 3.2.
     - This will create a plot with a single point at (2.5, 3.2).
   - Command: plot(2.5, 3.2)
3. **Create a Line Plot with Custom Color**
   - Use the plot() function with the following specifications:
     - The x-axis values are from 1 to 10, represented by 1:10.
     - Set the color of the plot to pink using col="pink".
     - Set the type of the plot to line (type="l").
   - Command: plot(1:10, col="pink", type="l")

4. **Create a Scatter Plot with Custom Size and Shape**
    - Use the plot() function with the following specifications:
        - The x-axis values are from 1 to 5, represented by 1:5.
        - Set the color of the points to Black using col="Black".
        - Set the size of the points to 2 times the default size using cex=2.
        - Set the shape of the points to 11 (which corresponds to a custom point symbol) using pch=11.
    - Command: plot(1:5, col="Black", cex=2, pch=11)
5. **End**

Aim: Write a program to illustrate Customizing and Saving to Graphs in R.

Description: The graph functions are possible to save graph using R programming code Specify files to save your image using functions such as jpeg( ), png( ) and pdf( ).

1.Saving graphs as jpeg:

We can create a jpeg file and then save a graph in it by using the jpeg( ) function.

Using the jpeg command, we create a jpeg file. R starts storing all output into the file. We use the barplot( ) function to create the graph. This stores the graph into the jpeg file. Finally, we use the dev.off( ) command to stop recording and save the file.

2.Saving graphs as png:

The png( ) function in R can create and save graphs in a png file.

Using the jpeg command, we create a jng file. R starts storing all output into the file. We use the barplot( ) function to create the graph. This stores the graph into the jng file. Finally, we usethe dev.off( ) command to stop recording and save the file.

3.Saving graphs as pdf:

We can use the pdf( ) function to save graphs as a pdf file.

Using the jpeg command, we create a jdf file. R starts storing all output into the file. We use the barplot( ) function to create the graph. This stores the graph into the jdf file. Finally, we use the dev.off( ) command to stop recording and save the file.

Algorithm:

1. **Start**
2. **Save Graph as JPEG**
    - Use jpeg() function to specify the output file format as a JPEG.
        - Provide the filename: "barplot.jpeg".
    - Create a bar plot using the barplot() function:
        - Set the data to airquality$Ozone.
        - Set the plot title as "Ozone concentration in air".
        - Label the x-axis as "ozone levels".
        - Set the color of the bars to "blue".
        - Set the bars to be horizontal using horiz=TRUE.
    - Use dev.off() to close the device and save the file.
    - Command:

      r
      Copy code

```
jpeg("barplot.jpeg")
barplot(airquality$Ozone,
        main="Ozone concentration in air",
        xlab='ozone levels',
        col="blue",
        horiz=TRUE)
dev.off()
```

3. **Save Graph as PNG**
    - Use png() function to specify the output file format as PNG.
        - Provide the filename: "barplot.png".
    - Create a bar plot with the same specifications as above.
    - Use dev.off() to close the device and save the file.
    - Command:

      ```
      r
      Copy code
      png("barplot.png")
      barplot(airquality$Ozone,
              main="Ozone concentration in air",
              xlab='ozone levels',
              col="blue",
              horiz=TRUE)
      dev.off()
      ```

4. **Save Graph as PDF**
    - Use pdf() function to specify the output file format as PDF.
        - Provide the filename: "barplot.pdf".
    - Create a bar plot with the same specifications as above.
    - Use dev.off() to close the device and save the file.
    - Command:

      ```
      r
      Copy code
      pdf("barplot.pdf")
      barplot(airquality$Ozone,

              main="Ozone concentration in air",
              xlab='ozone levels',
              col="blue",
              horiz=TRUE)
      dev.off()
      ```

5. **End**

Aim: Write a program to illustrate the data sets by applying statistical methods.

Description: Illustrating datasets through statistical methods in R-Programming involves applying various statistical techniques to gain insights into the data.

The **mtcars** dataset is a built-in dataset in R that contains measurements on 11 differentattributes for 32 different cars.

Algorithm:

1. **Start**
2. **Load the mtcars Dataset**
   - Assign the mtcars dataset to a variable data_cars.
   - Command:

     ```
     r
     data_cars = mtcars
     ```

3. **Check the Dimensions of the Dataset**
   - Use the dim() function to get the dimensions (number of rows and columns) of the dataset.
   - Command:

     ```
     r
     dim(data_cars)
     ```

   - The result will provide the number of rows and columns in data_cars.
4. **Check the Names of the Variables/Columns in the Dataset**
   - Use the names() function to retrieve the column names of the dataset.
   - Command:

     ```
     r
     names(data_cars)
     ```

   - This will list all the column names in the data_cars dataset.
5. **Get Row Names of the Dataset**
   - Use the rownames() function to retrieve the row names of the dataset.
   - Command:

     ```
     r
     rownames(data_cars)
     ```

   - This will display the names of the rows in data_cars.
6. **End**

Aim: Write a program to illustrate the data sets by data visualization.

Description: Data visualization is the technique used to deliver insights in data using visual cues such as graphs, charts, maps, and many others. This is useful as it helps in intuitive and easy understanding of the large quantities of data and there by make better decisions.

In R-programming data visualization as it offers flexibility and minimum required coding through its packages.

The popular data visualization tools that are available are Tableau, Plotly, R, Google Charts etc.

1.Plotting data sets:

Data Data Extraction: The next two lines extract the 'weight' and 'height' variables from the 'women' data frame. It assumes that there is a data frame named 'women' with columns named 'weight' and 'height'.

Plotting: The last line creates a scatter plot using the **plot** function, where 'height' is plotted against 'weight'. The arguments **xlab** and **ylab** specify the labels for the x-axis and y-axis

respectively. The **type = "o"** argument indicates that both points and lines should be plotted, with lines connecting the points. The **pch = 19** argument sets the point character to a solid circle, and **col = 'black'** sets the color of the points and lines to black.

The next two lines extract the 'weight' and 'height' variables from the 'women' data frame. It assumes that there is a data frame named 'women' with columns named 'weight' and 'height'.

Plotting: The last line creates a scatter plot using the **plot** function, where 'height' is plotted against 'weight'. The arguments **xlab** and **ylab** specify the labels for the x-axis and y-axis,

respectively. The **type = "o"** argument indicates that both points and lines should be plotted, with lines connecting the points. The **pch = 19** argument sets the point character to a solid circle, and **col = 'black'** sets the color of the points and lines to black.

2.Histogram:

A histogram is like a bar chart as it uses bars of varying height to represent data distribution. However, in a histogram values are grouped into consecutive intervals called bins. In a Histogram, continuous values are grouped and displayed in these bins whose size can be varied.

Example:
For a histogram, the parameter **xlim** can be used to specify the interval within which all valuesare to be displayed.
Another parameter **freq** when set to TRUE denotes the frequency of the various values in the histogram and when set to FALSE, the probability densities are represented on the y-axis such that they are of the histogram adds up to one.

Histograms are used in the following scenarios:

- To verify an equal and symmetric distribution of the data.
- To identify deviations from expected values.

3D Graphs in R:

Here we will use preps() function, This function is used to create 3D surfaces in perspective view. This function will draw perspective plots of a surface over the x–y plane.

Syntax: **persp(x, y, z)**

Parameter: This function accepts different parameters i.e. x, y and z where x and y are vectors defining the location along x- and y-axis. z-axis will be the height of the surface in thematrix z.

Return Value: persp( ) returns the viewing transformation matrix for projecting 3D coordinates (x, y, z) into the 2D plane using homogeneous 4D coordinates (x, y, z, t)

Algorithm:

1. **Load Required Libraries and Data:**
   - Load the women dataset and extract weight and height columns into variables.
2. **Scatter Plot of Weight vs. Height:**
   - Use the plot() function to plot a scatter plot where:
     - X-axis represents height.
     - Y-axis represents weight.
     - Customize the plot by setting labels for both axes (xlab, ylab), point type (pch), and color (col).
3. **Create Histogram for Daily Maximum Temperature:**
   - Load the airquality dataset.
   - Use hist() to create a histogram of airquality$Temp with the following settings:
     - Title for the plot (main).
     - X-axis label (xlab), which indicates temperature in Fahrenheit.
     - X-axis limits (xlim) to adjust the range.
     - Color (col) for the histogram bars.
     - Set frequency (freq=TRUE) to display counts.
4. **3D Surface Plot (Cone Function Example):**
   - Define a function cone(x, y) that calculates the square root of x^2 + y^2 (representing a cone).
   - Prepare x and y variables using the seq() function.
   - Use outer(x, y, cone) to compute the corresponding z values for the surface plot.
5. **Plot 3D Surface:**
   - Use the persp() function to plot a 3D surface with the following parameters:
     - main: Title of the 3D plot.
     - zlab: Label for the z-axis.
     - theta and phi: Adjust the angle of the 3D plot.
     - col: Define the color of the plot surface.
     - shade: Adjust the shading of the plot.

Viva Questions

1. What are R Graphics?
2. What is the plot( ) function in R?
3. How do you create a basic plot in R?
4. How can you customize graphs in R?
5. How do you import data from Excel and CSV files in R?
6. What is the ggplot2 **package, and why is it used?**
7**. How do you create a 3D plot in R?**