R Programming

Lab Manual

III YEAR- I SEM

R22

Department of CSE(Data Science)

2024-2025

Course Lab Faculty: HOD:

Mr T Shravan kumar Dr K Ramesh Rao

VISION AND MISSION OF THE DEPARTMENT

Vision:

To emerge as the Data Science Centre of Excellence in the region, offer technology services to industry, academia, and community

Mission:

- To provide modern infrastructure facilities to access, process, and analyze large-scale data sets.
- Continuous empowerment of faculty and students in the latest technological advancements in Data Science.
- To Strengthen industry connections, collaborate with academia, and develop projects to serve the community by large.

Program Outcomes and Program Specific Outcomes

PO 1	An ability to apply knowledge of computing, mathematics, science, and engineering fundamentals appropriate to the discipline.
PO 2	An Ability to Identify, formulate and analyze complex engineering problems, solving them by applying principles of mathematics and engineering sciences.
PO 3	An ability to design, effective and efficient solutions for problems to meet the requirements.
PO 4	Conduct investigations of complex problems using research-based knowledge and methods including design of experiments, analysis and interpretation of data, to provide valid conclusions
PO 5	An ability to adopt and apply modern technical concepts, tools and practices.
PO 6	An ability to analyze the impact of computing on individuals, organizations and society.
PO 7	Apply knowledge and skill set to develop solutions by considering environmental and sustainability constraints.
PO 8	An understanding of professional, ethical, legal, security and social issues and responsibilities with respect to technology and tools.
PO 9	Function effectively as an individual, and as a member or leader in diverse teams.
PO 10	An ability to communicate effectively.
PO 11	Ability to deliver cost effective solutions and contribute towards project management.
PO 12	Ability to engage in lifelong learning to abreast with modern technologies and practices.
PSO 1	**Professional Skills and Foundations of Software development:** Ability to analyze, design and implement applications by adopting the dynamic nature of Software developments.
PSO 2	**Applications of Computing and Research Ability**: Ability to use knowledge in cutting edge technologies in identifying research gaps and to render solutions with innovative ideas.

COURSE DESCRIPTION

Name of the Dept.: Data Science

Course Title	**R PROGRAMMING LAB**			
Course Code	**DS504PC**			
Regulation	R22			
Course Structure	Lectures	Tutorials	Practical	Credits
	0	0	2	1
Course Coordinator	**Mr T SHRAVAN KUMAR**			
Team of Instructors				

Course Objectives:

- Familiarize with R basic programming concepts, various data structures for handling datasets,
- various graph representations and Exploratory Data Analysis concepts

Course Outcomes:

- Setup R programming environment.
- Understand and use R – Data types and R – Data Structures.
- Develop programming logic using R – Packages.
- Analyze data sets using R – programming capabilities

Session Planner

EXP NO	Name of Experiments	Planned Date	Conducted Date	Action taken if not covered
1	Download and install R-Programming environment and install basic packages using install. packages() command in R.			
2	Learn all the basics of R-Programming (Data types, Variables, Operators etc,.)			
3	Write R command to i) Illustrate summation, subtraction, multiplication, and division operations on vectors using vectors. ii) Enumerate multiplication and division operations between matrices and vectors in R console			
4	Write R command to i) Illustrates the usage of Vector subsetting and Matrix subsetting ii) Write a program to create an array of 3×3 matrixes with 3 rows and 3 columns.			
5	Write an R program to draw i) Pie chart ii) 3D Pie Chart, iii) Bar Chart along with chart legend by considering suitable CSV file			
6	Create a CSV file having Speed and Distance attributes with 1000 records. Write R program to draw i) Box plots ii) Histogram iii) Line Graph iv) Multiple line graphs v) Scatter plot to demonstrate the relation between the cars speed and the distance.			
7	Implement different data structures in R (Vectors, Lists, Data Frames)			
8	Write an R program to read a csv file and analyze the data in the file using EDA (Explorative Data Analysis) techniques.			
9	Write an R program to illustrate Linear Regression and Multi linear Regression considering suitable CSV file			

SPHOORTHY ENGINEERING COLLEGE (AUTONOMOUS)

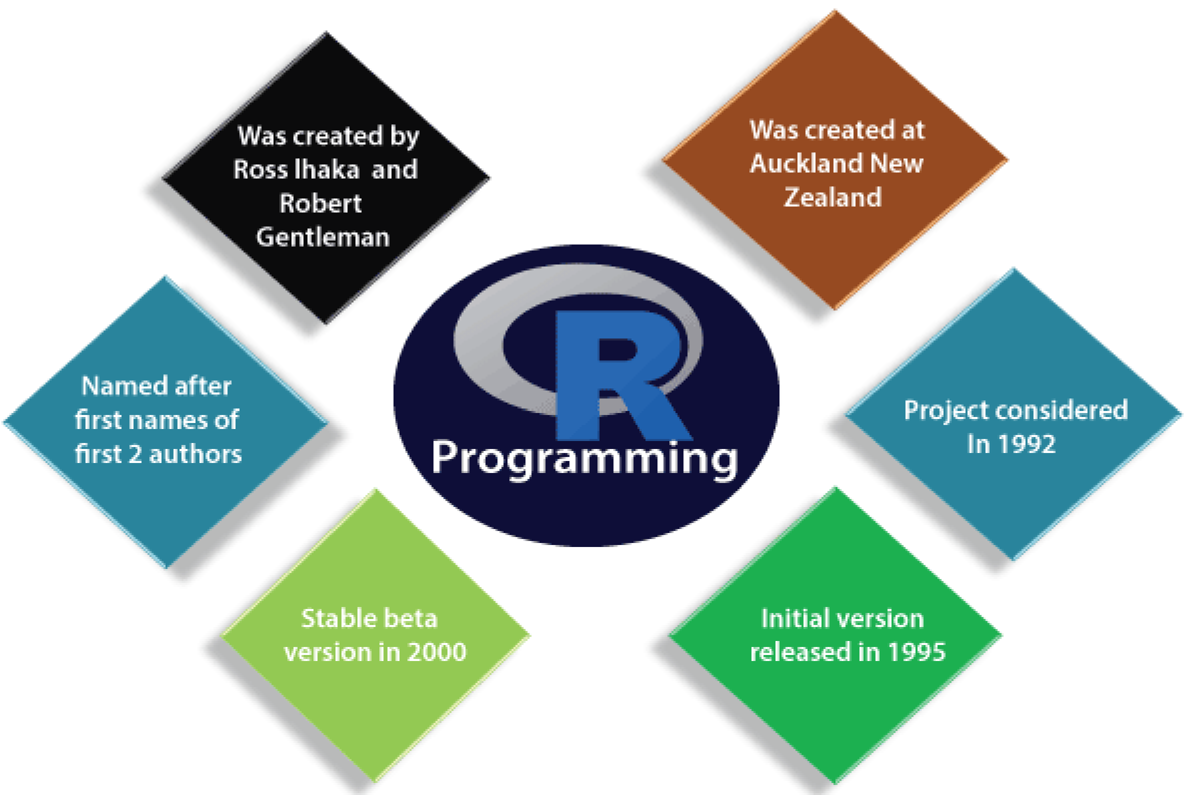
Passion Ignited @ Sphoorthy

Introduction:

"R is an interpreted computer programming language which was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand." The ***R Development Core Team*** currently develops R. It is also a software environment used to analyze **statistical information**, **graphical representation**, **reporting**, and **data modeling**. R is the implementation of the **S programming** language, which is combined with **lexical scoping semantics**.

History of R Programming:

The history of R goes back about 20-30 years ago. R was developed by Ross lhaka and Robert Gentleman in the University of Auckland, New Zealand, and the R Development Core Team currently develops it. This programming language name is taken from the name of both the developers. The first project was considered in 1992. The initial version was released in 1995, and in 2000, a stable beta version was released.



Features of R:

1. **Statistical Analysis**: R provides a wide array of statistical techniques such as linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, and more.
2. **Graphics**: R is well-known for its capabilities to produce high-quality graphical data representations. You can create everything from simple graphs to complex multi-panel plots.
3. **Packages**: R's functionality is greatly enhanced by its package system. Thousands of packages are available on CRAN (Comprehensive R Archive Network), allowing users to extend R's base functionality.
4. **Data Handling**: R provides extensive tools for data manipulation, cleaning, and aggregation.
5. **Programming**: R is a full-fledged programming language, supporting loops, conditional statements, and user-defined functions.

EXP 1: Download and install R-Programming environment and install basic packages using install. packages () command in R

<u>R-Environment Setup:</u>

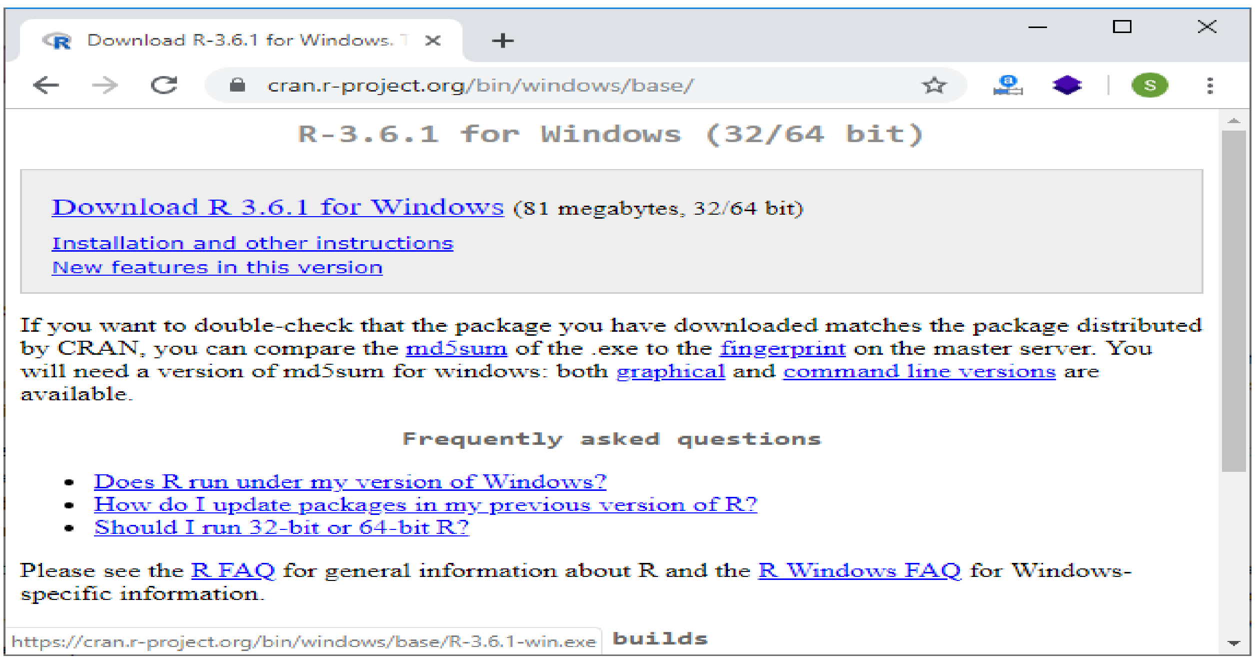
R programming is a very popular language and to work on that we have to install two things, i.e., R and RStudio. R and RStudio works together to create a project on R.

Install R in Windows

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

Step 1:

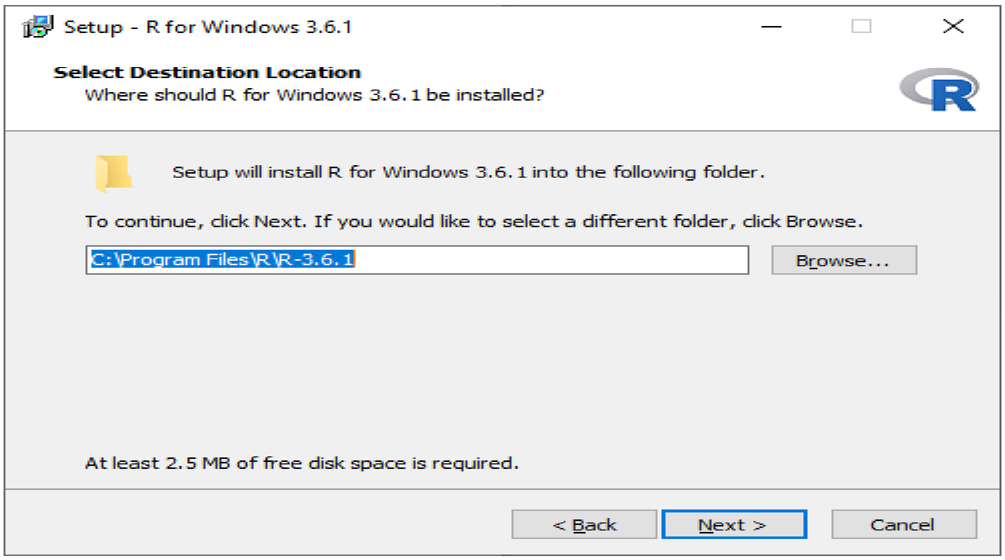
First, we have to download the R setup from https://cloud.r-project.org/bin/windows/base/.



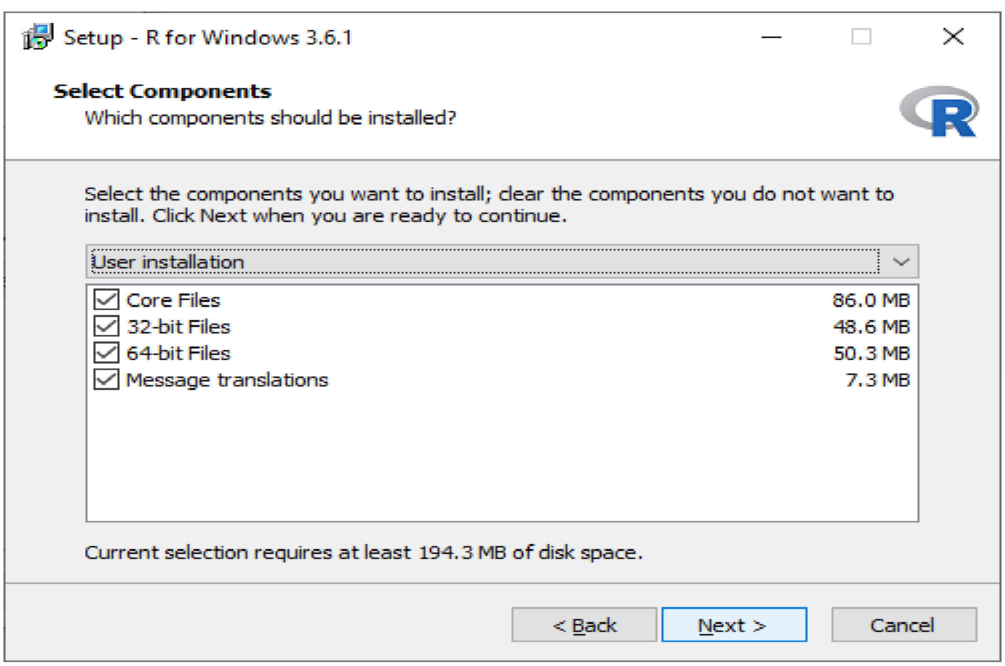
Step 2:

When we click on **Download R 3.6.1 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:

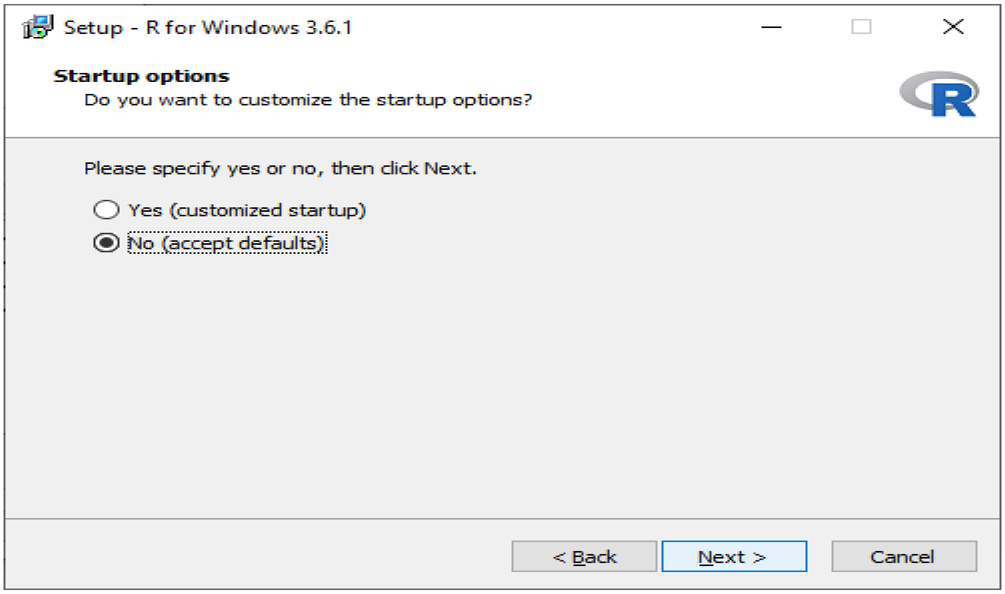
1) Select the path where we want to download the R and proceed to Next.



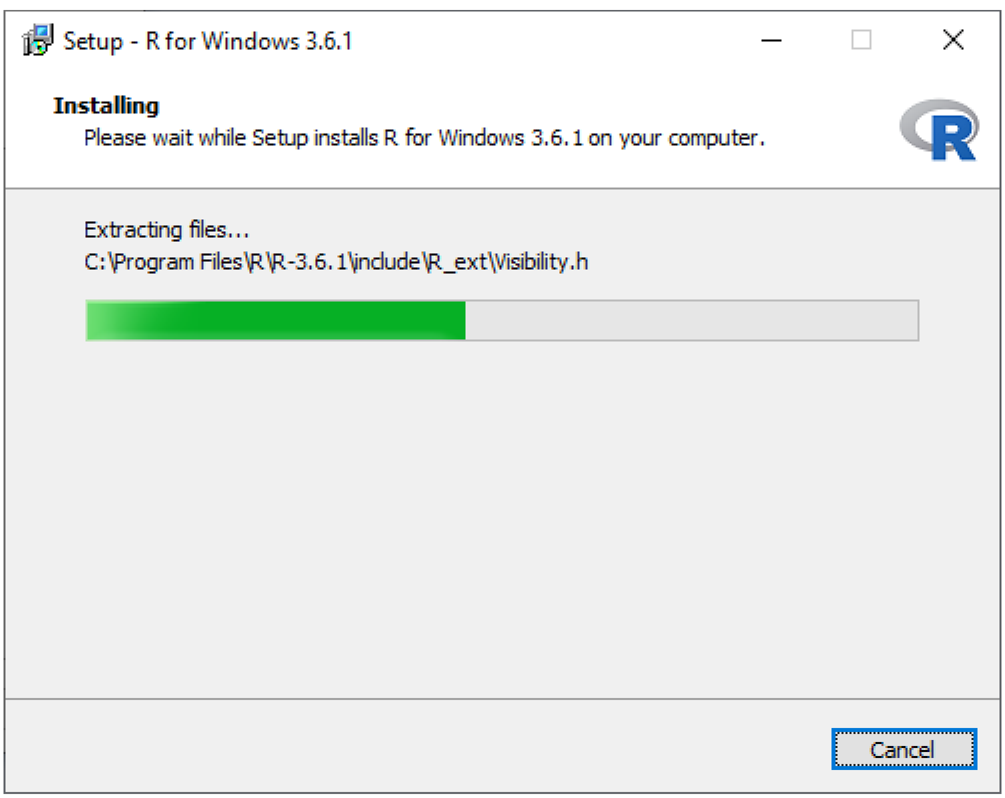
2) Select all components which we want to install, and then we will proceed to **Next**



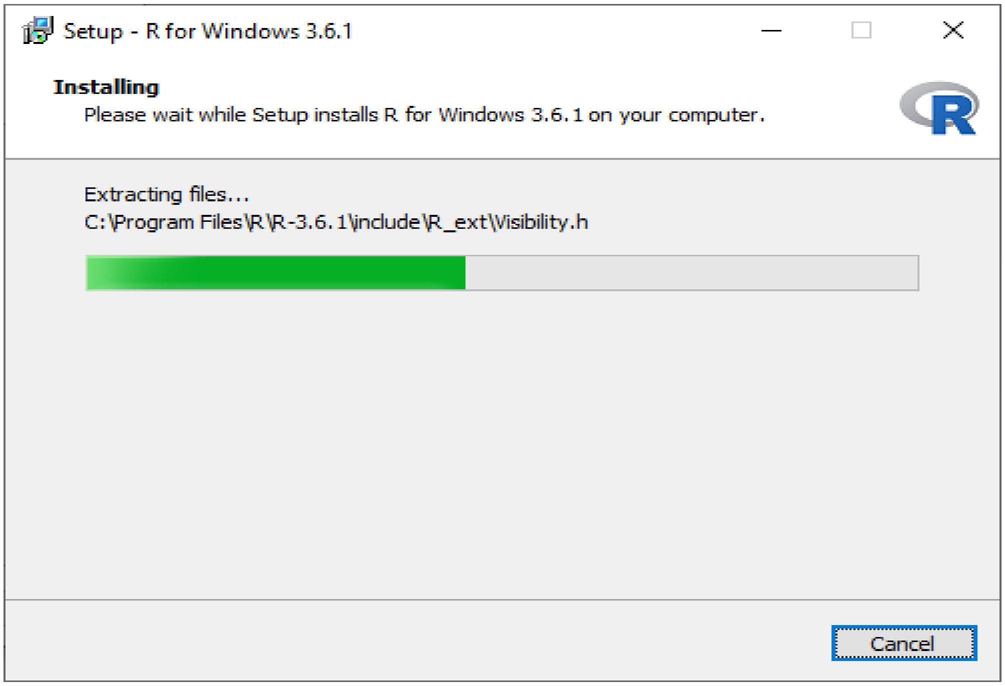
3) In the next step, we have to select either customized startup or accept the default, and then we proceed to **Next**.



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



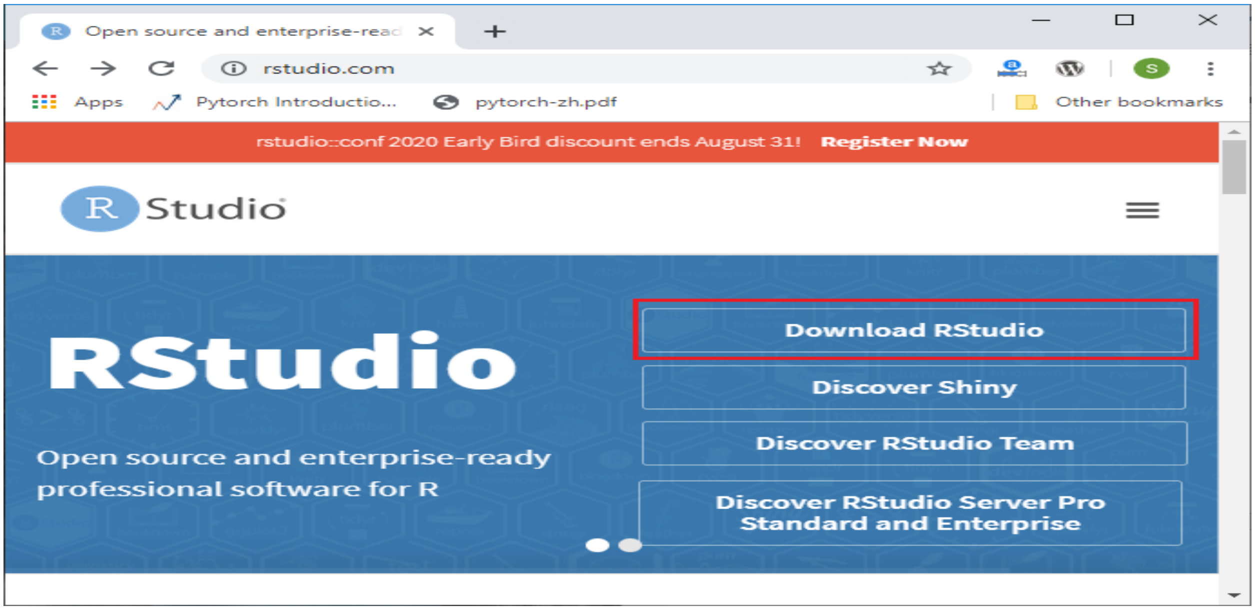
5) In the last, we will click on finish to successfully install R in our system.



<u>Installation of RStudio:</u>

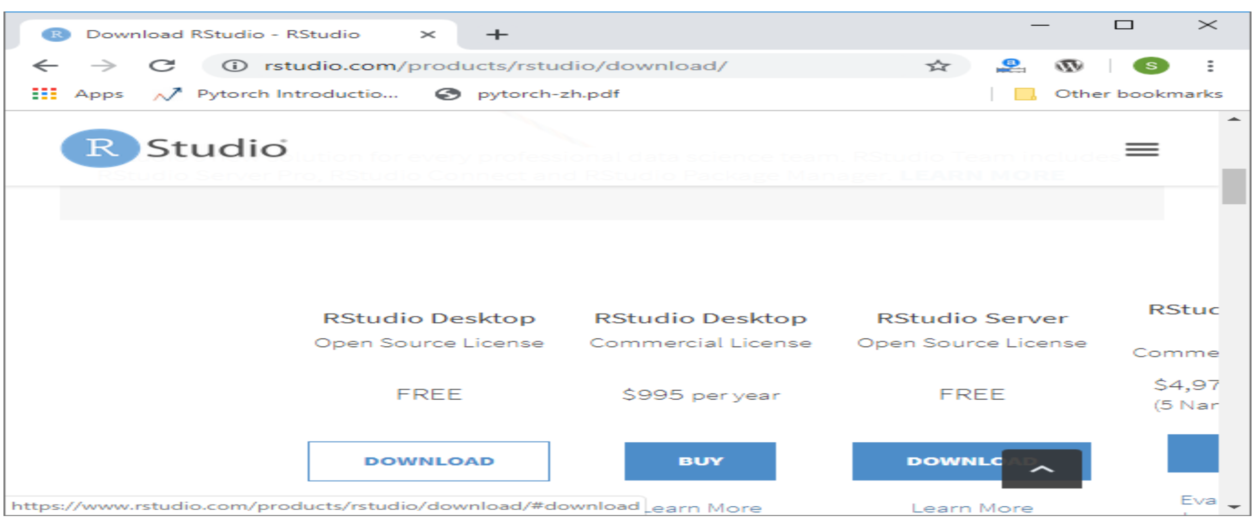
Step 1:

In the first step, we visit the RStudio official site and click on **Download RStudio**.



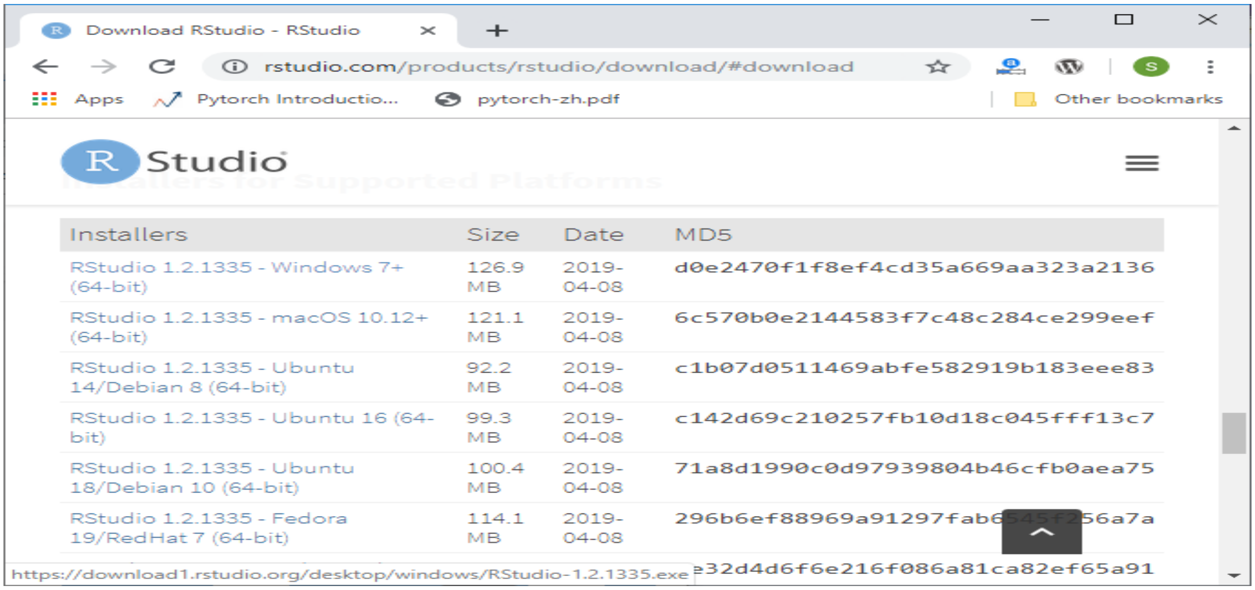
Step 2:

In the next step, we will select the RStudio desktop for open-source license and click on download.



Step 3:

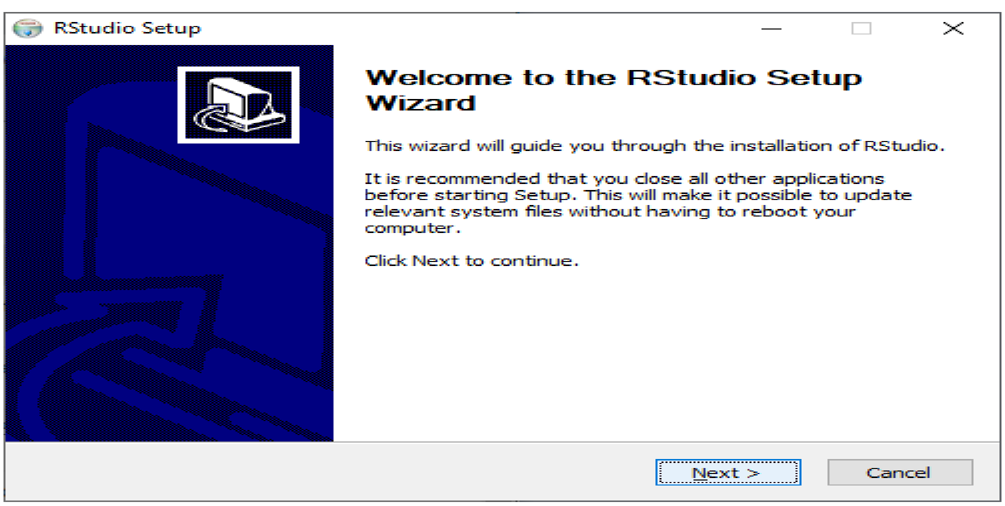
In the next step, we will select the appropriate installer. When we select the installer, our downloading of RStudio setup will start.



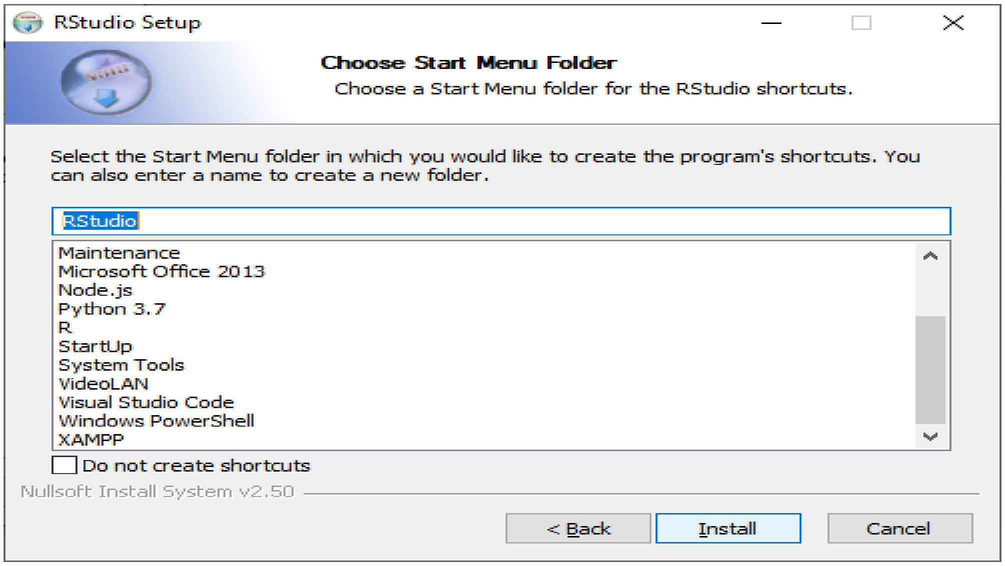
Step 4:

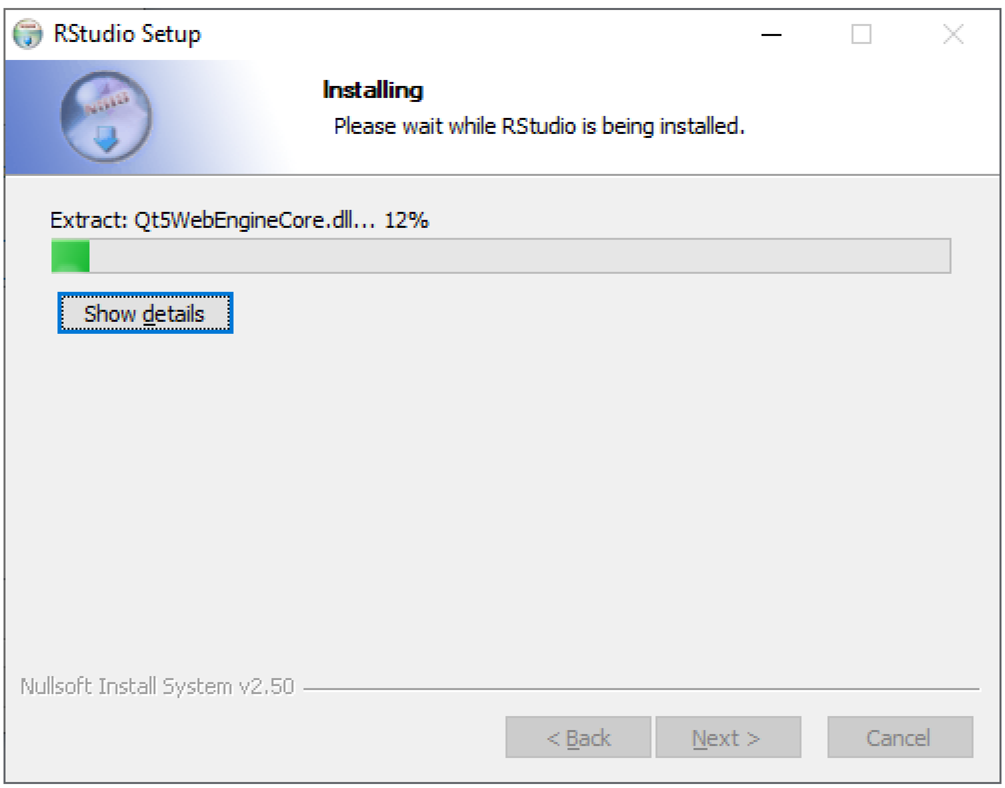
In the next step, we will run our setup in the following way:

1) Click on Next.

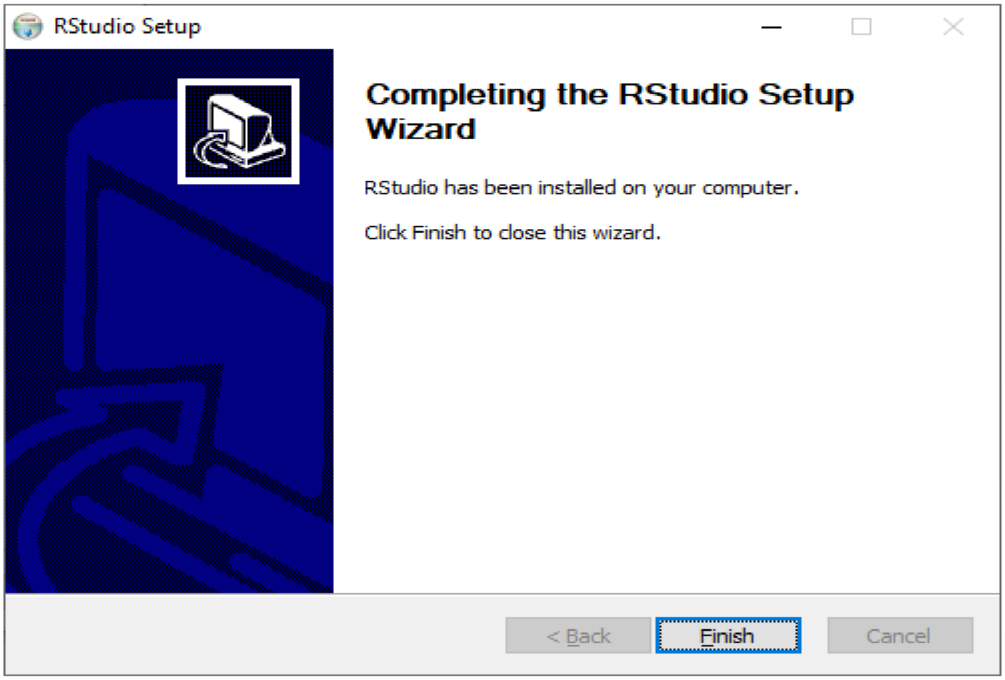


2) Click on Install.

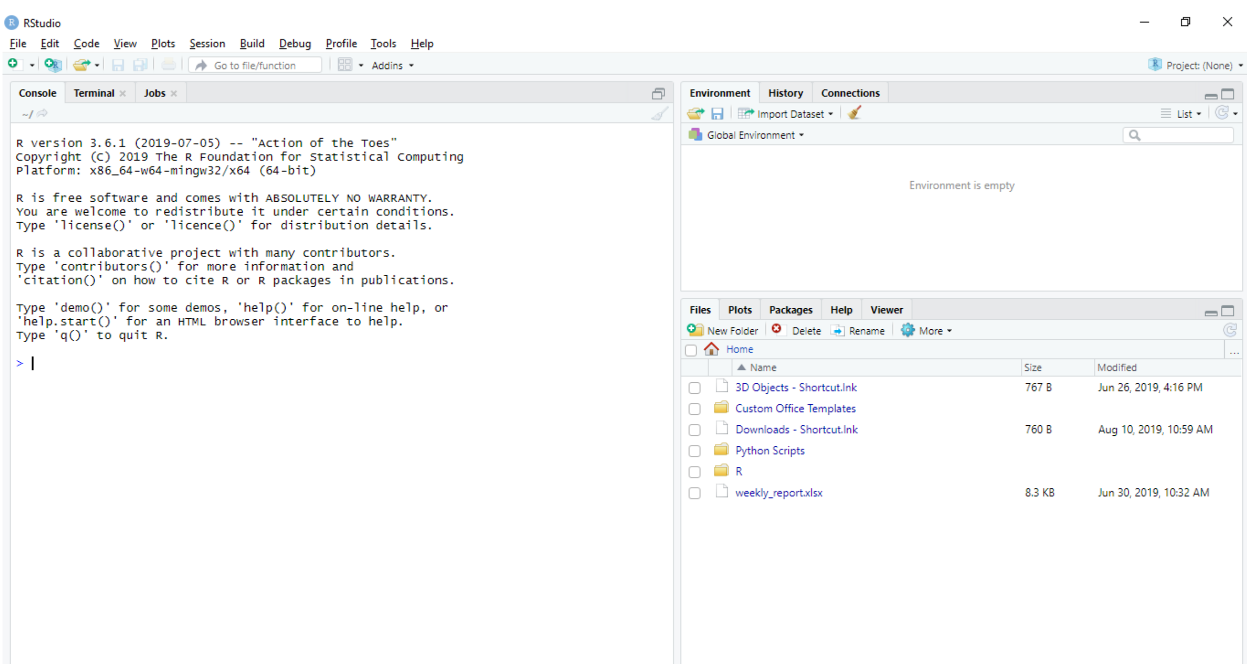




3) Click on finish.



4) RStudio is ready to work.



RStudio IDE:

RStudio Windows/Tabs	Location	Description
Console Window	Lower-left	The location where commands are entered and output is printed.
Source Tabs	Upper-left	Built-in test editor
Environment Tab	Upper-left	An interactive list of loaded R objects.
History Tab	Upper-left	List of keystrokes entered into the console.
Files Tab	Lower-right	File explorer to navigate C drive folders.
Plots Tab	Lower-right	Output location for plots.
Packages Tab	Lower-right	List of installed packages.
Help Tab	Lower-right	Output location for help commands and help search Window.
Viewer Tab	Lower-right	Advanced tab for local web content.

EXP-2. Learn all the basics of R-Programming (Data types, Variables, Operators etc,.)

1. Data Types in R

Understanding data types is fundamental in R programming. R primarily deals with the following data types:

- Numeric: Represents numbers, which can be integers or real numbers (with decimal points).
  - Example: 10, 42.5
- Integer: Represents whole numbers. An integer is denoted by appending L to a number.
  - Example: 10L, 42L
- Character (String): Represents text or string data.
  - Example: "Hello", "R Programming"
- Logical (Boolean): Represents truth values, typically TRUE or FALSE.
  - Example: TRUE, FALSE
- Complex: Represents complex numbers (numbers with real and imaginary parts).
  - Example: 2+3i
- Factor: Represents categorical data, often used in statistical modeling.
  - Example: factor(c("Male", "Female"))
- Date and Time: R provides classes for date (Date) and date-time (POSIXct and POSIXlt).
  - Example: as.Date("2024-08-26"), as.POSIXct("2024-08-26 10:00:00")

PROGRAM DATA TYPES:

```r
# Numeric
num <- 42.5
print(paste("Numeric:", num))
print(paste("Type:", typeof(num)))

# Integer
int <- 42L
print(paste("Integer:", int))
print(paste("Type:", typeof(int)))

# Character (String)
char <- "Hello, R!"
print(paste("Character:", char))
print(paste("Type:", typeof(char)))

# Logical (Boolean)
logi <- TRUE
print(paste("Logical:", logi))
print(paste("Type:", typeof(logi)))
```

```
# Complex
comp <- 2 + 3i
print(paste("Complex:", comp))
print(paste("Type:", typeof(comp)))

# Factor
fact <- factor(c("Male", "Female", "Male", "Female"))
print("Factor:")
print(fact)
print(paste("Type:", typeof(fact)))

# Date
date <- as.Date("2024-08-26")
print(paste("Date:", date))
print(paste("Type:", typeof(date)))

# Date-Time
datetime <- as.POSIXct("2024-08-26 14:00:00")
print(paste("Date-Time:", datetime))
print(paste("Type:", typeof(datetime)))
```

Output:

```
[1] "Numeric: 42.5"
[1] "Type: double"
[1] "Integer: 42"
[1] "Type: integer"
[1] "Character: Hello, R!"
[1] "Type: character"
[1] "Logical: TRUE"
[1] "Type: logical"
[1] "Complex: 2+3i"
[1] "Type: complex"
[1] "Factor:"
[1] Male   Female Male   Female
Levels: Female Male
[1] "Type: integer"
[1] "Date: 2024-08-26"
[1] "Type: double"
[1] "Date-Time: 2024-08-26 14:00:00"
[1] "Type: double"
```

2.Operators in R:

R supports various operators for performing calculations, comparisons, and logical operations.

Arithmetic Operators

- + : Addition
- - : Subtraction
- * : Multiplication
- / : Division
- ^ : Exponentiation
- %% : Modulus (remainder of division)
- %/% : Integer division

Comparison Operators

- == : Equal to
- != : Not equal to
- > : Greater than
- < : Less than
- >= : Greater than or equal to
- <= : Less than or equal to

Logical Operators

- & : Logical AND
- | : Logical OR
- ! : Logical NOT

Assignment Operators

- <- : Assign value to a variable (left assignment)
- -> : Assign value to a variable (right assignment)
- <<- : Global assignment operator, used within functions to assign values to variables in the global environment.

PROGRAM OPERATORS:

```
# Assigning values to variables
a <- 10
b <- 3

# 1. Arithmetic Operators
sum <- a + b
difference <- a - b
product <- a * b
quotient <- a / b
exponentiation <- a ^ b
remainder <- a %% b
int_division <- a %/% b

# Print the results of arithmetic operations
print(paste("Sum:", sum))
print(paste("Difference:", difference))
print(paste("Product:", product))
```

```
print(paste("Quotient:", quotient))
print(paste("Exponentiation:", exponentiation))
print(paste("Remainder:", remainder))
print(paste("Integer Division:", int_division))

# 2. Comparison Operators
is_equal <- a == b
is_not_equal <- a != b
is_greater <- a > b
is_less <- a < b
is_greater_equal <- a >= b
is_less_equal <- a <= b

# Print the results of comparison operations
print(paste("Is Equal:", is_equal))
print(paste("Is Not Equal:", is_not_equal))
print(paste("Is Greater:", is_greater))
print(paste("Is Less:", is_less))
print(paste("Is Greater or Equal:", is_greater_equal))
print(paste("Is Less or Equal:", is_less_equal))
```

```
# 3. Logical Operators
x <- TRUE
y <- FALSE

logical_and <- x & y
logical_or <- x | y
logical_not <- !x

# Print the results of logical operations
print(paste("Logical AND (x & y):", logical_and))
print(paste("Logical OR (x | y):", logical_or))
print(paste("Logical NOT (!x):", logical_not))

# 4. Assignment Operators
z <- 42     # Left assignment
42 -> w     # Right assignment

# Print the values of z and w
print(paste("Value of z:", z))
print(paste("Value of w:", w))
```

Output:

```
[1] "Sum: 13"
[1] "Difference: 7"
[1] "Product: 30"
[1] "Quotient: 3.33333333333333"
[1] "Exponentiation: 1000"
[1] "Remainder: 1"
[1] "Integer Division: 3"
[1] "Is Equal: FALSE"
[1] "Is Not Equal: TRUE"
[1] "Is Greater: TRUE"
[1] "Is Less: FALSE"
[1] "Is Greater or Equal: TRUE"
[1] "Is Less or Equal: FALSE"
[1] "Logical AND (x & y): FALSE"
[1] "Logical OR (x | y): TRUE"
[1] "Logical NOT (!x): FALSE"
[1] "Value of z: 42"
[1] "Value of w: 42"
```

3. Variables in R:

Variables in R are used to store data values. You can assign a value to a variable using the assignment operator <- or =.

PROGRAM VARIABLES:

```
# 1. Variable Assignment

# Assigning numeric values to variables

x <- 10

y <- 20


# Assigning character (string) values to variables

name <- "Alice"

greeting <- "Hello, world!"


# Assigning logical (boolean) values to variables

is_active <- TRUE

is_admin <- FALSE


# 2. Performing Operations with Variables

# Arithmetic operations with numeric variables

sum <- x + y

difference <- x - y

product <- x * y

quotient <- x / y
```

```
# Combining strings using paste
full_greeting <- paste(greeting, name)

# Logical operations with boolean variables
is_member <- is_active & is_admin  # AND operation
can_access <- is_active | is_admin  # OR operation

# 3. Updating Variables
# Incrementing x by 5
x <- x + 5

# Changing the value of name
name <- "Bob"

# 4. Print the Variables and Results
print(paste("Value of x:", x))
print(paste("Value of y:", y))
print(paste("Sum of x and y:", sum))
print(paste("Difference of x and y:", difference))
print(paste("Product of x and y:", product))
print(paste("Quotient of x and y:", quotient))
print(paste("Full Greeting:", full_greeting))
```

```
print(paste("Is Member (AND):", is_member))
print(paste("Can Access (OR):", can_access))
print(paste("Updated name:", name))
```

Output:

```
[1] "Value of x: 15"
[1] "Value of y: 20"
[1] "Sum of x and y: 30"
[1] "Difference of x and y: -10"
[1] "Product of x and y: 200"
[1] "Quotient of x and y: 0.5"
[1] "Full Greeting: Hello, world! Alice"
[1] "Is Member (AND): FALSE"
[1] "Can Access (OR): TRUE"
[1] "Updated name: Bob"
```

EXP-3:

I) Illustrate summation, subtraction, multiplication, and division operations on vectors using vectors.

PROGRAM:

```
# Defining two vectors
vector1 <- c(10, 20, 30, 40, 50)
vector2 <- c(5, 4, 3, 2, 1)

# 1. Summation of Vectors
vector_sum <- vector1 + vector2
print("Summation of vectors:")
print(vector_sum)

# 2. Subtraction of Vectors
vector_difference <- vector1 - vector2
print("Subtraction of vectors:")
print(vector_difference)

# 3. Multiplication of Vectors
vector_product <- vector1 * vector2
print("Multiplication of vectors:")
print(vector_product)
```

```
# 4. Division of Vectors

vector_quotient <- vector1 / vector2

print("Division of vectors:")

print(vector_quotient)
```

Output:

```
[1] "Summation of vectors:"
[1] 15 24 33 42 51

[1] "Subtraction of vectors:"
[1] 5 16 27 38 49

[1] "Multiplication of vectors:"
[1] 50 80 90 80 50

[1] "Division of vectors:"
[1] 2.0 5.0 10.0 20.0 50.0
```

ii) Enumerate multiplication and division operations between matrices and vectors in R console

PROGRAM:

```
# Define a matrix (3x3) and a vector (length 3)
matrix_A <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3, byrow = TRUE)
vector_B <- c(1, 2, 3)

# 1. Element-wise Multiplication (Matrix and Vector)
# The vector is recycled along the rows of the matrix
elementwise_multiplication <- matrix_A * vector_B
print("Element-wise Multiplication of Matrix and Vector:")
print(elementwise_multiplication)

# 2. Element-wise Division (Matrix and Vector)
# The vector is recycled along the rows of the matrix
elementwise_division <- matrix_A / vector_B
print("Element-wise Division of Matrix and Vector:")
print(elementwise_division)

# 3. Matrix Multiplication (Matrix and Vector)
# This requires the vector to be treated as a column vector (length matches columns of the matrix)
# Use the %*% operator
matrix_multiplication <- matrix_A %*% vector_B
```

print("Matrix Multiplication of Matrix and Vector:")

print(matrix_multiplication)

Output:

```
[1] "Element-wise Multiplication of Matrix and Vector:"
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    8   10   12
[3,]   21   24   27

[1] "Element-wise Division of Matrix and Vector:"
         [,1]     [,2]     [,3]
[1,] 1.000000 2.000000 3.000000
[2,] 2.000000 2.500000 3.000000
[3,] 2.333333 2.666667 3.000000

[1] "Matrix Multiplication of Matrix and Vector:"
     [,1]
[1,]   14
[2,]   32
[3,]   50
```

EXP-4:

Write R command to

i) Illustrates the usage of Vector subsetting and Matrix subsetting

Subsetting is a powerful technique in R that allows you to extract specific elements, rows, or columns from vectors and matrices. Below are R commands that illustrate how to subset vectors and matrices.

Program on Vector Subsetting:

```
# Define a vector
vector_A <- c(10, 20, 30, 40, 50)

# 1. Subsetting by Index
subset1 <- vector_A[2]  # Extract the 2nd element
subset2 <- vector_A[3:5]  # Extract elements from the 3rd to the 5th position

# 2. Subsetting by Logical Vector
logical_subset <- vector_A[vector_A > 25]  # Extract elements greater than 25

# 3. Subsetting by Negative Index
negative_subset <- vector_A[-1]  # Exclude the 1st element

# Print the results of vector subsetting
print("Subset by Index (2nd element):")
print(subset1)
print("Subset by Index (3rd to 5th elements):")
```

print(subset2)

print("Subset by Logical Vector (elements > 25):")

print(logical_subset)

print("Subset by Negative Index (excluding 1st element):")

print(negative_subset)

Output:

[1] "Subset by Index (2nd element):"

[1] 20

[1] "Subset by Index (3rd to 5th elements):"

[1] 30 40 50

[1] "Subset by Logical Vector (elements > 25):"

[1] 30 40 50

[1] "Subset by Negative Index (excluding 1st element):"

[1] 20 30 40 50

Program on Matrix Subsetting:

```
# Define a 3x3 matrix

matrix_B <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3, byrow = TRUE)


# 1. Subsetting by Row and Column Index

single_element <- matrix_B[2, 3]  # Extract the element in the 2nd row, 3rd column

row_subset <- matrix_B[1, ]       # Extract the entire 1st row

column_subset <- matrix_B[, 2]    # Extract the entire 2nd column


# 2. Subsetting by Logical Matrix

logical_matrix <- matrix_B > 4    # Create a logical matrix where elements > 4

logical_subset_matrix <- matrix_B[logical_matrix]  # Extract elements greater than 4


# Print the results of matrix subsetting

print("Single Element (2nd row, 3rd column):")

print(single_element)

print("Row Subset (1st row):")

print(row_subset)

print("Column Subset (2nd column):")

print(column_subset)

print("Logical Subset (elements > 4):")

print(logical_subset_matrix)
```

Output:

```
[1] "Single Element (2nd row, 3rd column):"
[1] 6

[1] "Row Subset (1st row):"
[1] 1 2 3

[1] "Column Subset (2nd column):"
[1] 2 5 8

[1] "Logical Subset (elements > 4):"
[1] 5 6 7 8 9
```

ii) Write a program to create an array of 3×3 matrixes with 3 rows and 3 columns.

Program:

```
# Define the data for the array
data <- c(1:27)  # A sequence of numbers from 1 to 27

# Create the array
# The array has 3 rows, 3 columns, and 3 layers (3 matrices of 3x3)
array_3x3 <- array(data, dim = c(3, 3, 3))

# Print the array
print("Array of 3x3 matrices:")
print(array_3x3)
```

Output:

```
[1] "Array of 3x3 matrices:"
, , 1

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

```
, , 2

    [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18

, , 3

    [,1] [,2] [,3]
[1,]   19   22   25
[2,]   20   23   26
[3,]   21   24   27
```

EXP-7: Implement different data structures in R (Vectors, Lists, Data Frames)

Data Structures in R Programming:

A data structure is a particular way of organizing data in a computer so that it can be used effectively. The idea is to reduce the space and time complexities of different tasks. Data structures in R programming are tools for holding multiple values.

R's base data structures are often organized by their dimensionality (1D, 2D, or nD) and whether they're homogeneous (all elements must be of the identical type) or heterogeneous (the elements are often of various types). This gives rise to the six data types which are most frequently utilized in data analysis.

The most essential data structures used in R include:

- Vectors
- Lists
- Dataframes
- Matrices
- Arrays
- Factors

Vectors:

A **vector** is a basic data structure which plays an important role in R programming.

In R, a sequence of elements which share the same data type is known as vector. A vector supports logical, integer, double, character, complex, or raw data type. The elements which are contained in vector known as **components** of the vector. We can check the type of vector with the help of the **typeof()** function.

The length is an important property of a vector. A vector length is basically the number of elements in the vector, and it is calculated with the help of the **length()** function.

Creating and Naming Vectors:

In R, vectors are one of the most fundamental data structures. You can create and name vectors in various ways. Here's a brief guide:

Creating Vectors

1. **Using the c() Function:** The c() function combines values into a vector.

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

2. **Using the seq() Function:** The seq() function generates a sequence of numbers.
   my_vector <- seq(1, 10, by=2)   # Generates 1, 3, 5, 7, 9

3. **Using the rep() Function:** The rep() function repeats elements of a vector.
   my_vector <- rep(1:3, times=3)   # Repeats the sequence 1, 2, 3 three times

Naming Vectors

You can assign names to the elements of a vector using the names() function. This is useful for identifying elements within the vector.

1. **Assigning Names**:

   my_vector <- c(10, 20, 30)

   names(my_vector) <- c("first", "second", "third")

2. **Accessing Named Elements**: You can access elements by their names.
   my_vector["second"]   # Returns 20

PROGRAM:

```
# Create a vector
my_vector <- c(5, 10, 15)

# Name the elements
names(my_vector) <- c("low", "medium", "high")

# Print the vector
print(my_vector)


# Output: low  medium   high
#           5      10      15


# Access an element by name
print(my_vector["medium"])


# Output: 10
```

Introduction to Lists

A list in R is an ordered collection of elements, where each element can be of a different type or structure. Lists are particularly useful when dealing with datasets that are heterogeneous in nature.

Key Characteristics of a List:

- Heterogeneous Elements: Lists can store elements of different types (e.g., numeric, character, logical, and even other lists).
- Indexing: Elements in a list can be accessed by their index or by their names (if the list elements are named).
- Nested Structure: Lists can contain other lists, allowing for complex, nested data structures.

PROGRAM:

Creating a List

You can create a list in R using the list() function. Here's how you can create a simple list:

Example 1: Creating a Basic List

```
# Creating a list with different types of elements
my_list <- list(
  name = "Alice",
  age = 25,
  height = 5.5,
  is_student = TRUE
)

# Displaying the list
print(my_list)
```

Output:

```
name
[1] "Alice"

age
[1] 25
```

height
[1] 5.5

is_student
[1] TRUE

Example 2: Creating a List with Vectors

You can also create lists that contain vectors, which is useful for storing related data together.

```
# Creating a list with vectors
my_vector_list <- list(
  numbers = c(1, 2, 3, 4, 5),
  letters = c("A", "B", "C"),
  logicals = c(TRUE, FALSE, TRUE)
)

# Displaying the list
print(my_vector_list)
```

Output:

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

letters
[1] "A" "B" "C"
```

Accessing Elements in a List

You can access elements in a list using the $ operator, double square brackets [[ ]], or single square brackets [ ].

Accessing Elements by Name

```
# Accessing the 'name' element
print(my_list$name)

# Accessing the 'age' element
print(my_list$age)
```

Output:
[1] "Alice"
[1] 25

Accessing Elements by Index

```
# Accessing the first element
print(my_list[[1]])

# Accessing the third element
print(my_list[[3]])
```

Output:
[1] "Alice"
[1] 5.5

Introduction to Data Frame:

A data frame in R is one of the most commonly used data structures, particularly for data analysis. It is similar to a table or spreadsheet in that it organizes data into rows and columns, where each column can hold different types of data (e.g., numeric, character, factor). Data frames are powerful because they allow you to work with and manipulate structured datasets efficiently.

Key Features of a Data Frame

1. **Tabular Structure**: A data frame is essentially a list of vectors of equal length, where each vector forms a column.
2. **Mixed Data Types**: Unlike matrices, which can only hold one type of data, data frames can have different types of data in different columns (e.g., numeric, character, factor).
3. **Row and Column Names**: Data frames have row names (often just numbers) and column names (which describe the data in each column).
4. **Data Manipulation**: R provides various functions for data manipulation, such as subsetting, filtering, and transforming data frames.

PROGRAM:

Creating a Data Frame:

You can create a data frame using the data.frame() function.

Example: Simple Data Frame:

```
# Creating a simple data frame
df <- data.frame(
  Name = c("Alice", "Bob", "Charlie"),
  Age = c(25, 30, 35),
  Gender = factor(c("Female", "Male", "Male")),
```

```
 Height = c(5.5, 6.0, 5.8)
)
# Displaying the data frame
print(df)
```

Output:

```
    Name Age Gender Height
1  Alice  25 Female   5.5
2    Bob  30   Male   6.0
3 Charlie  35   Male   5.8
```

Accessing Data in a Data Frame

You can access data in a data frame using several methods:

1. **By Column Name:**

```
df$Name  # Access the 'Name' column
```

2. **By Row and Column Indices:**

```
df[1, ]  # Access the first row
df[, 2]  # Access the second column (Age)
df[1, 2] # Access the element in the first row, second column
```

3. **By Column Name and Row Index:**

```
df["Name"]   # Access the 'Name' column
df[1, "Age"] # Access the 'Age' value for the first row
```

Manipulating Data Frames:

You can easily manipulate data frames in R by adding or removing rows and columns, filtering data, and more.

Adding a New Column

```
# Adding a new column 'Weight'
df$Weight <- c(120, 150, 180)
print(df)
```

Output:

```
__Name Age Gender Height Weight
1  Alice  25 Female   5.5    120
2    Bob  30   Male   6.0    150
3 Charlie  35   Male   5.8    180
```

subsetting of Data Frames:

You can subset data frames based on conditions:

```
# Subsetting rows where Age is greater than 28
subset_df <- df[df$Age > 28, ]
print(subset_df)
```

Output:

```
__Name Age Gender Height Weight
2    Bob  30   Male   6.0    150
3 Charlie  35   Male   5.8    180
```

EXP-5: Write an R program to draw i) Pie chart ii) 3D Pie Chart, iii) Bar Chart along with chart legend by considering suitable CSV file

PROGRAM:

```
install.packages("plotrix")

# Load necessary library

library(plotrix) # for 3D pie chart


# Load data from CSV file

data <- read.csv("C:/Users/SPHOORTHY/Downloads/Courses.csv", header = TRUE)


# Assume the CSV has columns: "Category" and "Value"

categories <- data$Category

values <- data$Value


# i) Create a Pie Chart

pie(values, labels = categories, main = "Pie Chart", col =
rainbow(length(categories)))


# ii) Create a 3D Pie Chart

pie3D(values, labels = categories, main = "3D Pie Chart", col =
rainbow(length(categories)), explode = 0.1)
```

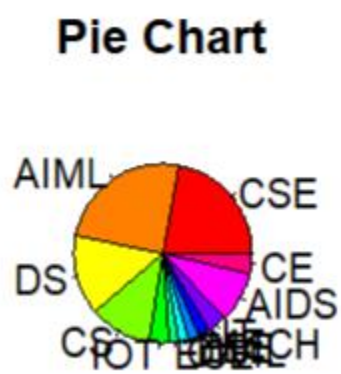
 # iii) Create a Bar Chart with Legend

barplot(values, names.arg = categories, main = "Bar Chart", col = rainbow(length(categories)))

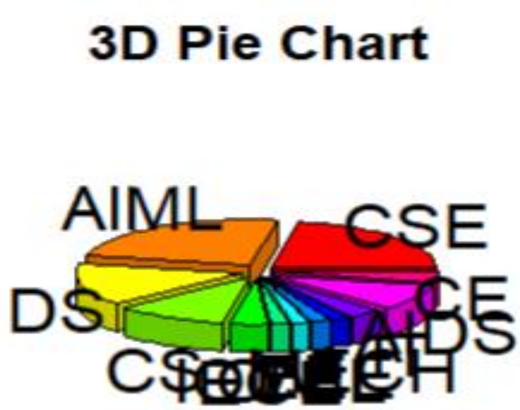
legend("topright", legend = categories, fill = rainbow(length(categories)), title = "Categories")

Output:

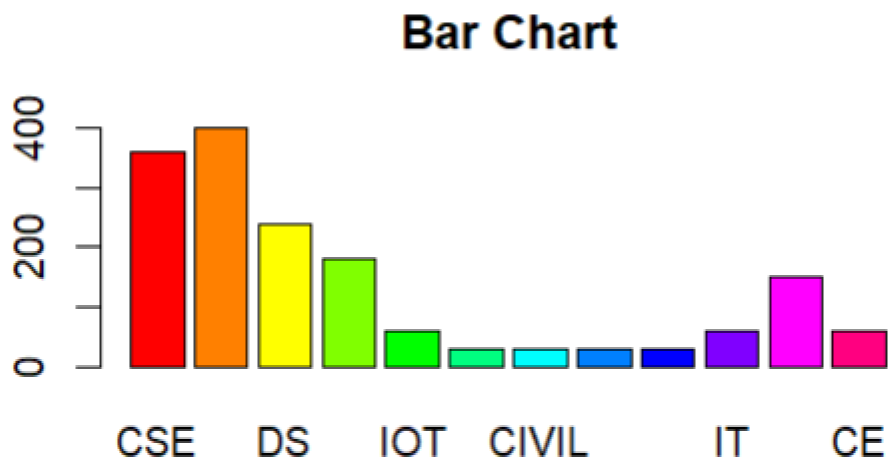
Pie chart:



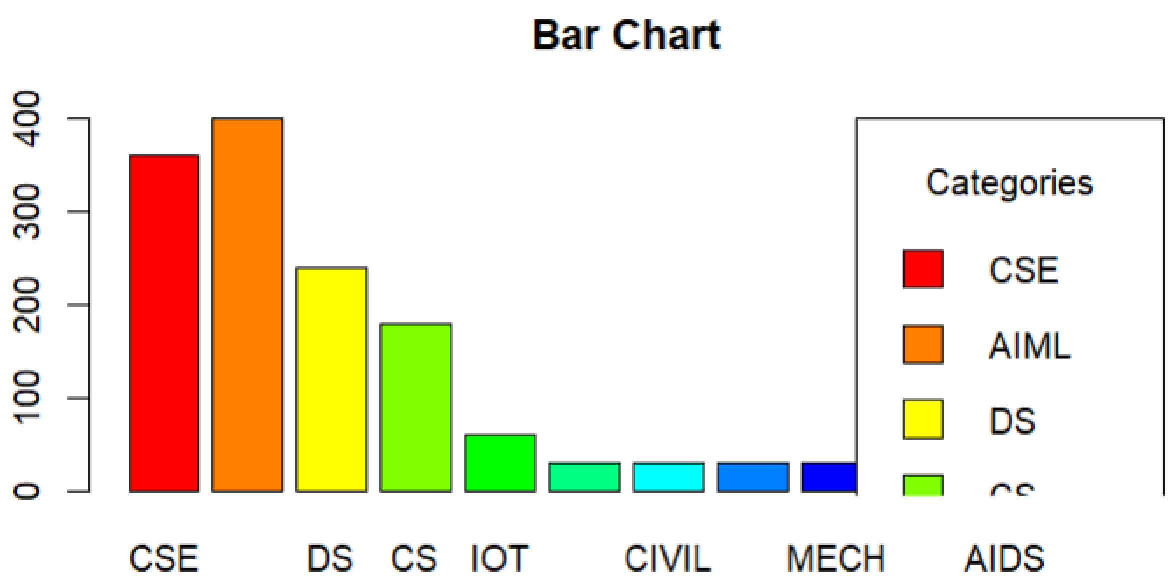
3D Pie chart:



Bar Chart:



Bar chart with Legend:



EXP-6: Create a CSV file having Speed and Distance attributes with 1000 records. Write R program to

draw i) Box plots

ii) Histogram

iii) Line Graph

iv) Multiple line graphs

v) Scatter plot

to demonstrate the relation between the cars speed and the distance.

PROGRAM:

```
# Create 1000 records of Speed and Distance

set.seed(123) # Set seed for reproducibility

speed <- runif(1000, min = 10, max = 100) # Random speed between 10 and 100

distance <- speed * runif(1000, min = 0.5, max = 2) # Distance depending on speed with some random factor


# Combine into a data frame

data <- data.frame(Speed = speed, Distance = distance)


# Write to CSV file

write.csv(data, "speed_distance_data.csv", row.names = FALSE)

# Load data from CSV file

data <- read.csv("speed_distance_data.csv", header = TRUE)
```

```
# Extract speed and distance
speed <- data$Speed
distance <- data$Distance

# i) Box Plots
boxplot(speed, distance,
        names = c("Speed", "Distance"),
        main = "Box Plots for Speed and Distance",
        col = c("lightblue", "lightgreen"))

# ii) Histogram
hist(speed,
     main = "Histogram of Speed",
     xlab = "Speed",
     col = "lightblue",
     border = "black")

hist(distance,
     main = "Histogram of Distance",
     xlab = "Distance",
     col = "lightgreen",
     border = "black")
```

 # iii) Line Graph

```
plot(speed, type = "l", col = "blue", xlab = "Index", ylab = "Speed",
    main = "Line Graph of Speed")
```

 # iv) Multiple Line Graphs

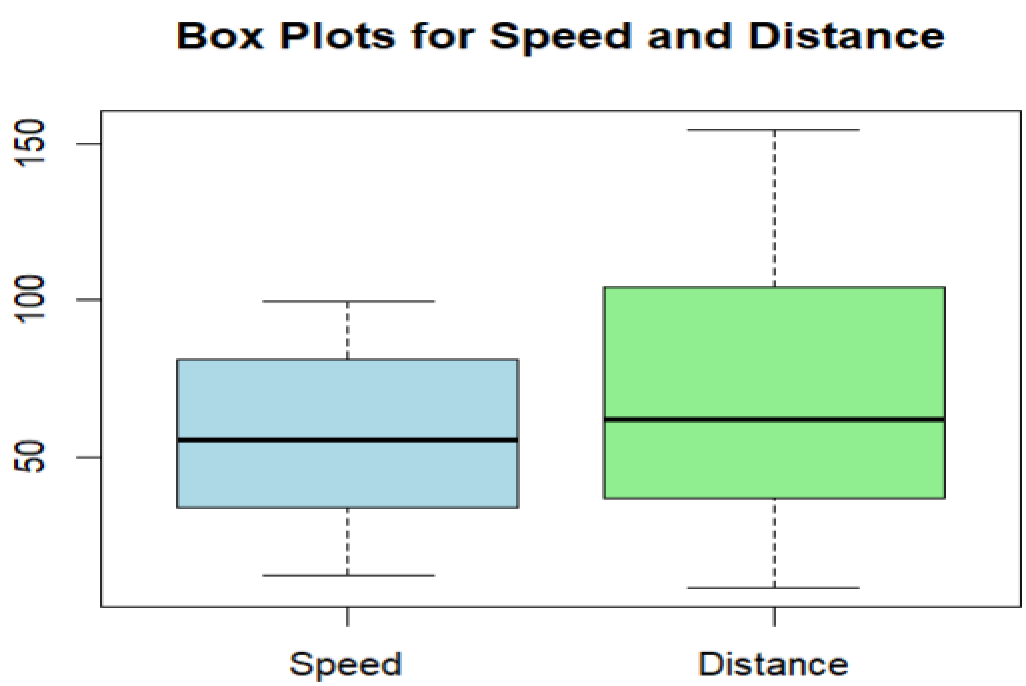
```
plot(speed, type = "l", col = "blue", xlab = "Index", ylab = "Values",
    main = "Multiple Line Graphs for Speed and Distance")
lines(distance, type = "l", col = "red")
legend("topright", legend = c("Speed", "Distance"),
      col = c("blue", "red"), lty = 1)
```

 # v) Scatter Plot

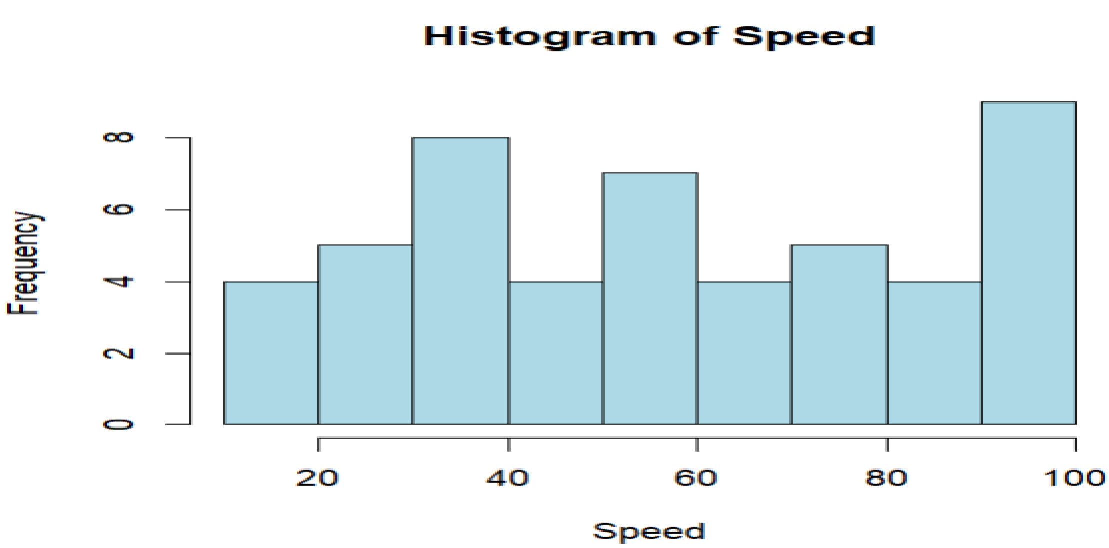
```
plot(speed, distance,
    main = "Scatter Plot of Speed vs Distance",
    xlab = "Speed",
    ylab = "Distance",
    col = "darkgreen",
    pch = 19)
```

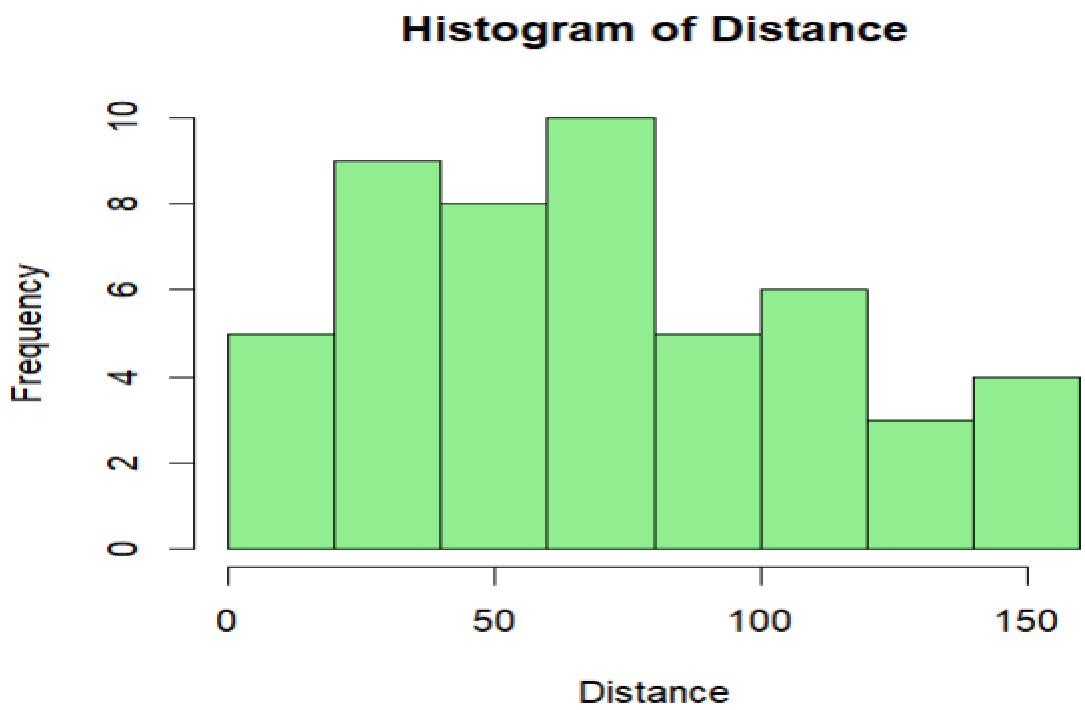
Output:

Box Plots:

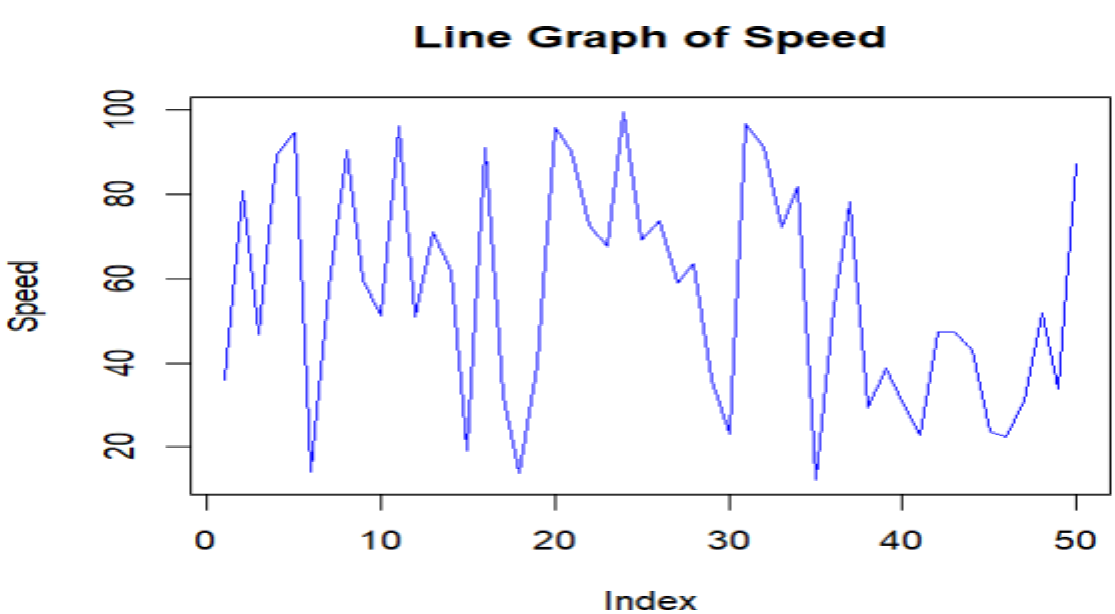


Histogram:

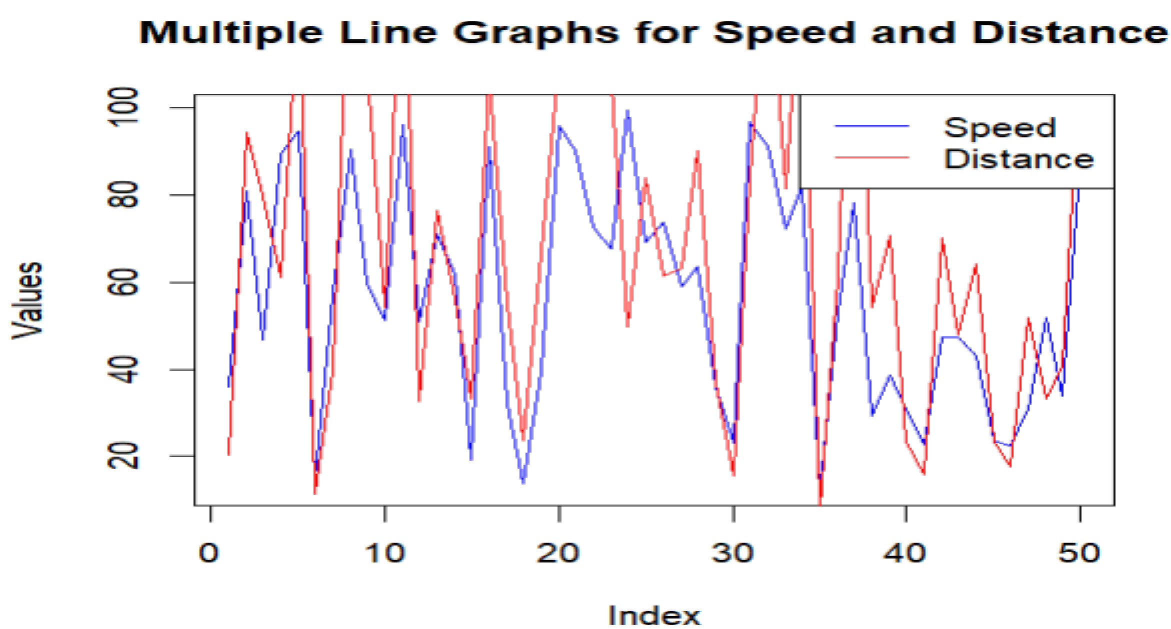



Histogram of Distance
Frequency
0
2
4
6
8
10
0
50
100
150
Distance

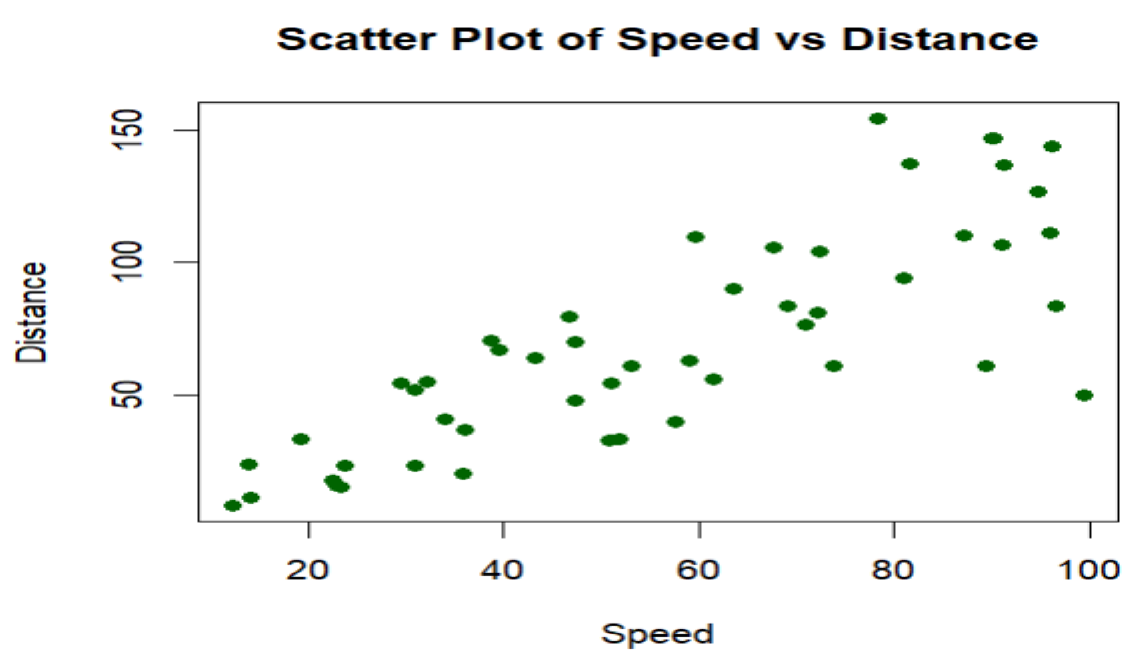
Line Graph:


Line Graph of Speed
Speed
20
40
60
80
100
0
10
20
30
40
50
Index

Multiple Line Graphs:



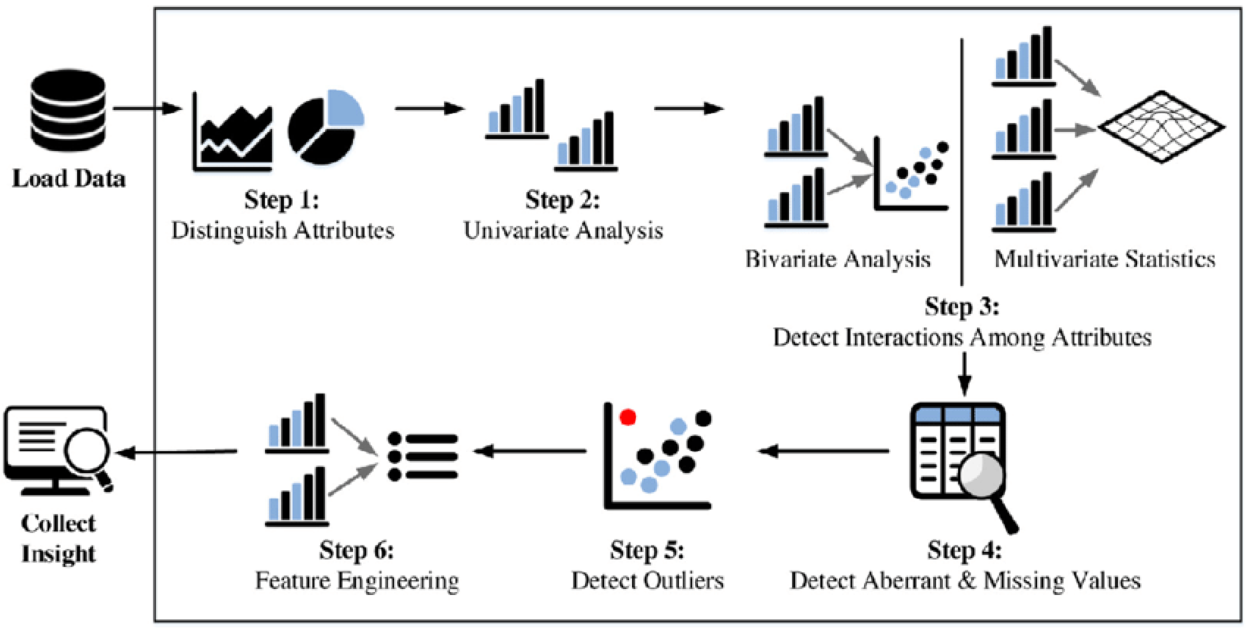
Scatter Plot:



EXP-8: Write an R program to read a csv file and analyze the data in the file using EDA (Explorative Data Analysis) techniques

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns,to spot anomalies,to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

It is a good practice to understand the data first and try to gather as many insights from it. EDA is all about making sense of data in hand,before getting them dirty with it.



PROGRAM:

```
library('ggvis')

library('tidyverse')

library('ggplot2')

bike_buyers = read.csv("C:/Users/SPHOORTHY/Downloads/bike_buyers.csv", header=T, na.strings='')

head(bike_buyers)
```

A data.frame: 6 × 13

	ID	Marital.Status	Gender	Income	Children	Education	Occupation	Home.Owner	Cars	Commute.Distance	Region	Age	Purchas
	<int>	<fct>	<fct>	<int>	<int>	<fct>	<fct>	<fct>	<int>	<fct>	<fct>	<int>	<fct>
1	12496	Married	Female	40000	1	Bachelors	Skilled Manual	Yes	0	0-1 Miles	Europe	42	No
2	24107	Married	Male	30000	3	Partial College	Clerical	Yes	1	0-1 Miles	Europe	43	No
3	14177	Married	Male	80000	5	Partial College	Professional	No	2	2-5 Miles	Europe	60	No
4	24381	Single	NA	70000	0	Bachelors	Professional	Yes	1	5-10 Miles	Pacific	41	Yes
5	25597	Single	Male	30000	0	Bachelors	Clerical	No	0	0-1 Miles	Europe	36	Yes
6	13507	Married	Female	10000	2	Partial College	Manual	Yes	0	1-2 Miles	Europe	50	No

class(bike_buyers)

str(bike_buyers)

```
'data.frame':   1000 obs. of  13 variables:
 $ ID              : int  12496 24107 14177 24381 25597 13507 27974 19364 22155 19280 ...
 $ Marital.Status  : Factor w/ 2 levels "Married","Single": 1 1 1 2 2 1 2 1 NA 1 ...
 $ Gender          : Factor w/ 2 levels "Female","Male": 1 2 2 NA 2 1 2 2 2 2 ...
 $ Income          : int  40000 30000 80000 70000 30000 10000 160000 40000 20000 NA ...
 $ Children        : int  1 3 5 0 0 2 2 1 2 2 ...
 $ Education       : Factor w/ 5 levels "Bachelors","Graduate Degree",..: 1 4 4 1 1 4 3 1 5 4 ...
 $ Occupation      : Factor w/ 5 levels "Clerical","Management",..: 5 1 4 4 1 3 2 5 1 3 ...
 $ Home.Owner      : Factor w/ 2 levels "No","Yes": 2 2 1 2 1 2 NA 2 2 2 ...
 $ Cars            : int  0 1 2 1 0 0 4 0 2 1 ...
 $ Commute.Distance: Factor w/ 5 levels "0-1 Miles","1-2 Miles",..: 1 1 4 5 1 2 1 1 5 1 ...
 $ Region          : Factor w/ 3 levels "Europe","North America",..: 1 1 1 3 1 1 3 1 3 1 ...
 $ Age             : int  42 43 60 41 36 50 33 43 58 NA ...
 $ Purchased.Bike  : Factor w/ 2 levels "No","Yes": 1 1 1 2 2 1 2 2 1 2 ...
```

summary(bike_buyers)

```
       ID         Marital.Status    Gender          Income           Children
 Min.   :11000   Married:535     Female:489   Min.   : 10000   Min.   :0.00
 1st Qu.:15291   Single :458     Male  :500   1st Qu.: 30000   1st Qu.:0.00
 Median :19744   NA's   :  7     NA's  : 11   Median : 60000   Median :2.00
 Mean   :19966                                Mean   : 56268   Mean   :1.91
 3rd Qu.:24471                                3rd Qu.: 70000   3rd Qu.:3.00
 Max.   :29447                                Max.   :170000   Max.   :5.00
                                              NA's   :6        NA's   :8
               Education             Occupation   Home.Owner       Cars
 Bachelors          :306   Clerical      :177   No  :314   Min.   :0.000
 Graduate Degree    :174   Management    :173   Yes :682   1st Qu.:1.000
 High School        :179   Manual        :119   NA's:  4   Median :1.000
 Partial College    :265   Professional  :276              Mean   :1.455
 Partial High School: 76   Skilled Manual:255              3rd Qu.:2.000
                                                           Max.   :4.000
                                                           NA's   :9
```

```
 Commute.Distance          Region           Age          Purchased.Bike
0-1 Miles :366    Europe       :300   Min.   :25.00   No :519
1-2 Miles :169    North America:508   1st Qu.:35.00   Yes:481
10+ Miles :111    Pacific      :192   Median :43.00
2-5 Miles :162                        Mean   :44.18
5-10 Miles:192                        3rd Qu.:52.00
                                      Max.   :89.00
                                      NA's   :8
```

levels(bike_buyers$Gender)

1. **'Female'**
2. **'Male'**

bike_buyers$Marital.Status <- as.factor(bike_buyers$Marital.Status)

bike_buyers$Gender <- as.factor(bike_buyers$Gender)

bike_buyers$Home.Owner <- as.factor(bike_buyers$Home.Owner)

bike_buyers$Purchased.Bike <- as.factor(bike_buyers$Purchased.Bike)

str(bike_buyers)

```
'data.frame':   1000 obs. of  13 variables:
 $ ID              : int  12496 24107 14177 24381 25597 13507 27974 19364 22155 19280 ...
 $ Marital.Status  : Factor w/ 2 levels "Married","Single": 1 1 1 2 2 1 2 1 NA 1 ...
 $ Gender          : Factor w/ 2 levels "Female","Male": 1 2 2 NA 2 1 2 2 2 2 ...
 $ Income          : int  40000 30000 80000 70000 30000 10000 160000 40000 20000 NA ...
 $ Children        : int  1 3 5 0 0 2 2 1 2 2 ...
 $ Education       : Factor w/ 5 levels "Bachelors","Graduate Degree",..: 1 4 4 1 1 4 3 1 5 4 ...
 $ Occupation      : Factor w/ 5 levels "Clerical","Management",..: 5 1 4 4 1 3 2 5 1 3 ...
 $ Home.Owner      : Factor w/ 2 levels "No","Yes": 2 2 1 2 1 2 NA 2 2 2 ...
 $ Cars            : int  0 1 2 1 0 0 4 0 2 1 ...
 $ Commute.Distance: Factor w/ 5 levels "0-1 Miles","1-2 Miles",..: 1 1 4 5 1 2 1 1 5 1 ...
 $ Region          : Factor w/ 3 levels "Europe","North America",..: 1 1 1 3 1 1 3 1 3 1 ...
 $ Age             : int  42 43 60 41 36 50 33 43 58 NA ...
 $ Purchased.Bike  : Factor w/ 2 levels "No","Yes": 1 1 1 2 2 1 2 2 1 2 ...
```

colSums(is.na(bike_buyers))

ID

0

Marital.Status

7

Gender

11

Income

6

Children

8

Education

0

Occupation

0

Home.Owner

4

Cars

9

Commute.Distance

0

Region

0

Age

8

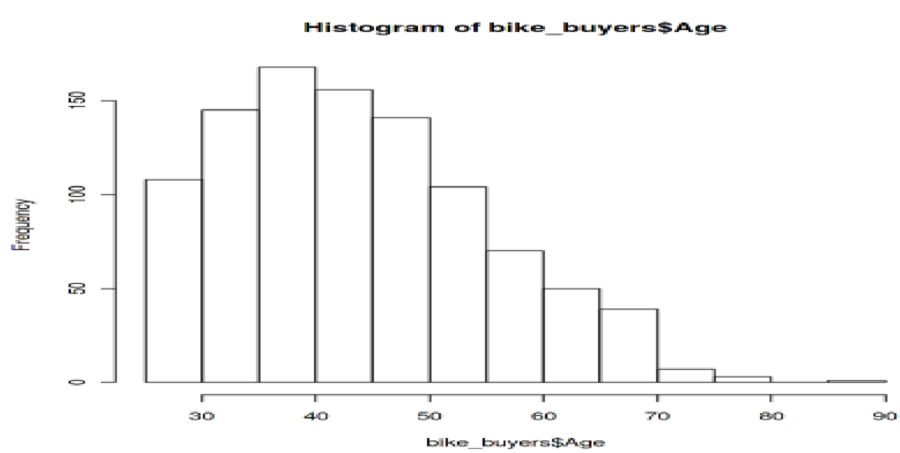
Purchased.Bike

0

summary(bike_buyers)

```
       ID         Marital.Status    Gender        Income          Children
 Min.   :11000   Married:535    Female:489   Min.   : 10000   Min.   :0.00
 1st Qu.:15291   Single :458    Male  :500   1st Qu.: 30000   1st Qu.:0.00
 Median :19744   NA's   :  7    NA's  : 11   Median : 60000   Median :2.00
 Mean   :19966                               Mean   : 56268   Mean   :1.91
 3rd Qu.:24471                               3rd Qu.: 70000   3rd Qu.:3.00
 Max.   :29447                               Max.   :170000   Max.   :5.00
                                             NA's   :6        NA's   :8
              Education            Occupation  Home.Owner     Cars
 Bachelors          :306   Clerical      :177   No  :314   Min.   :0.000
 Graduate Degree    :174   Management    :173   Yes :682   1st Qu.:1.000
 High School        :179   Manual        :119   NA's:  4   Median :1.000
 Partial College    :265   Professional  :276              Mean   :1.455
 Partial High School: 76   Skilled Manual:255              3rd Qu.:2.000
                                                           Max.   :4.000
```

hist(bike_buyers$Age)



Dealing with NA values¶

Since, the distribution of Income and Age is left-skewed. We will impute median values

median(na.omit((bike_buyers$Income)))

median(na.omit((bike_buyers$Age)))

60000

43

bike_buyers_clean <- bike_buyers

colSums(is.na(bike_buyers_clean))

ID

0

Marital.Status

7

Gender

11

Income

6

Children

8

Education

0

Occupation

0

Home.Owner

4

Cars

9

Commute.Distance

0

Region

0

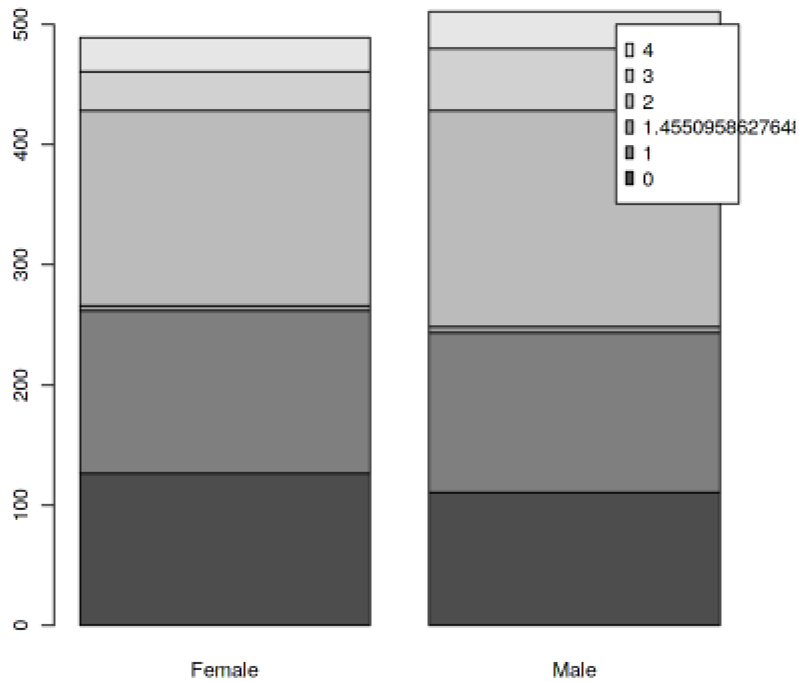
Age

8

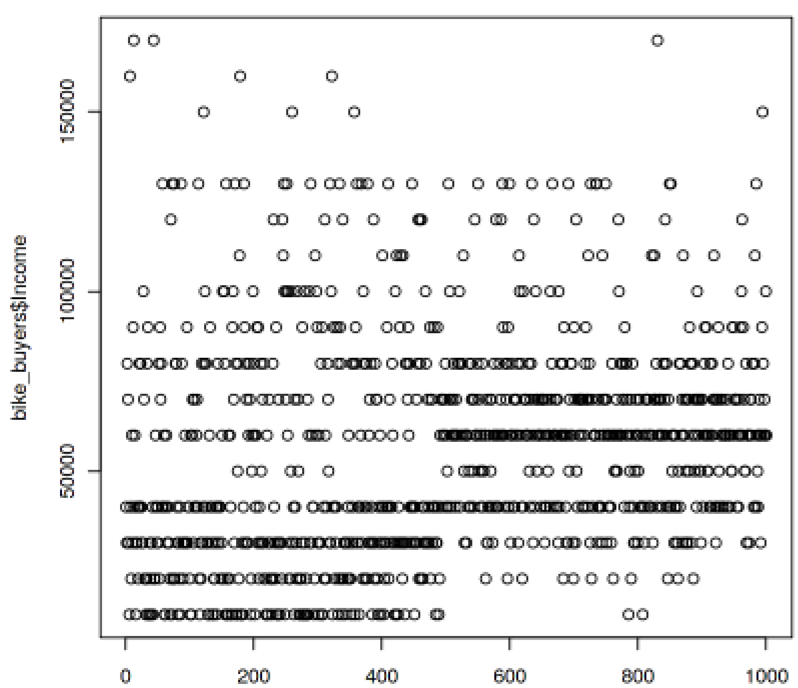
Purchased.Bike

0

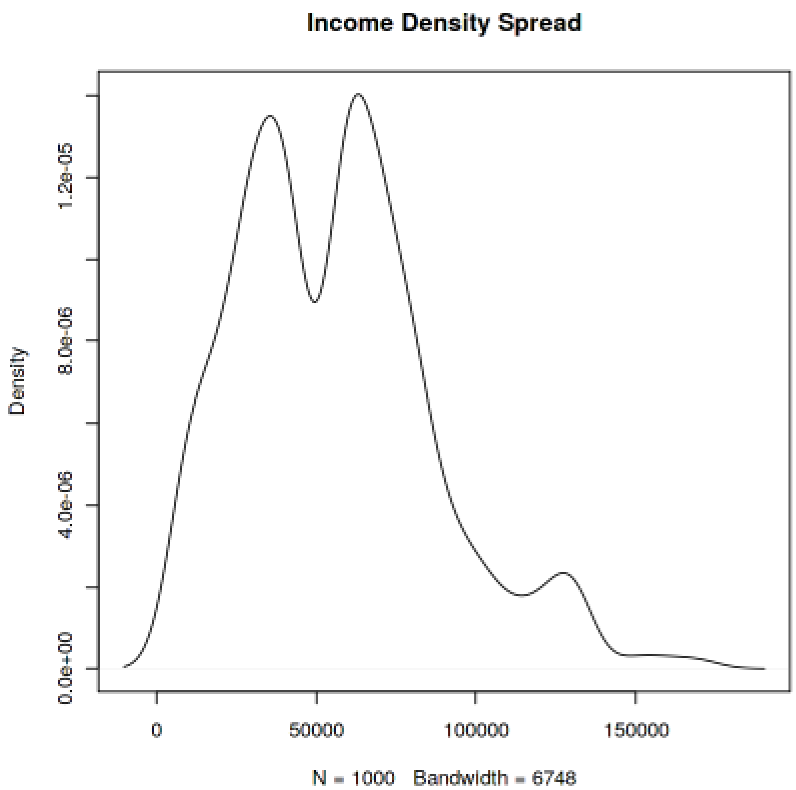
```
counts <- table(bike_buyers$Cars, bike_buyers$Gender)
barplot(counts, main = '',
      xlab="Number of Gears",
      legend = rownames(counts))
```



plot(bike_buyers$Income, type= "p")

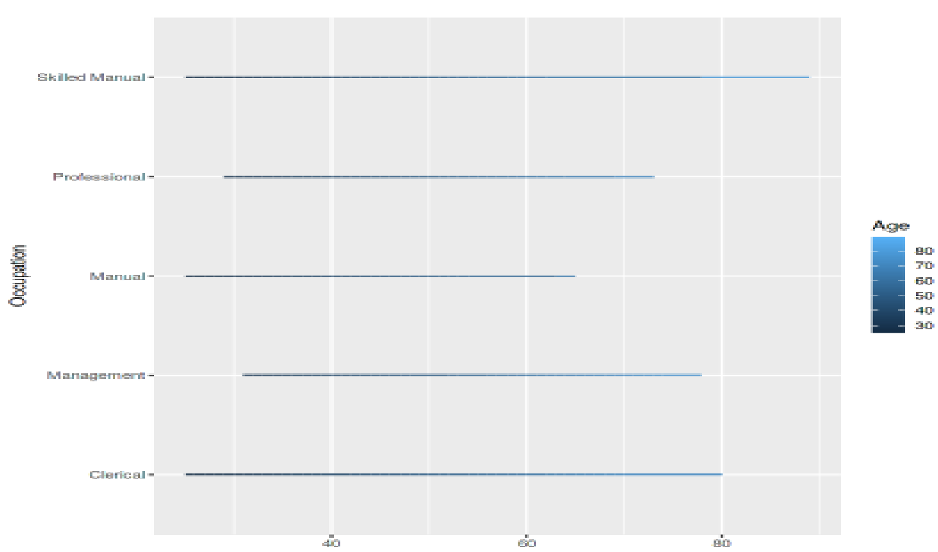


plot(density(bike_buyers$Income), main='Income Density Spread')



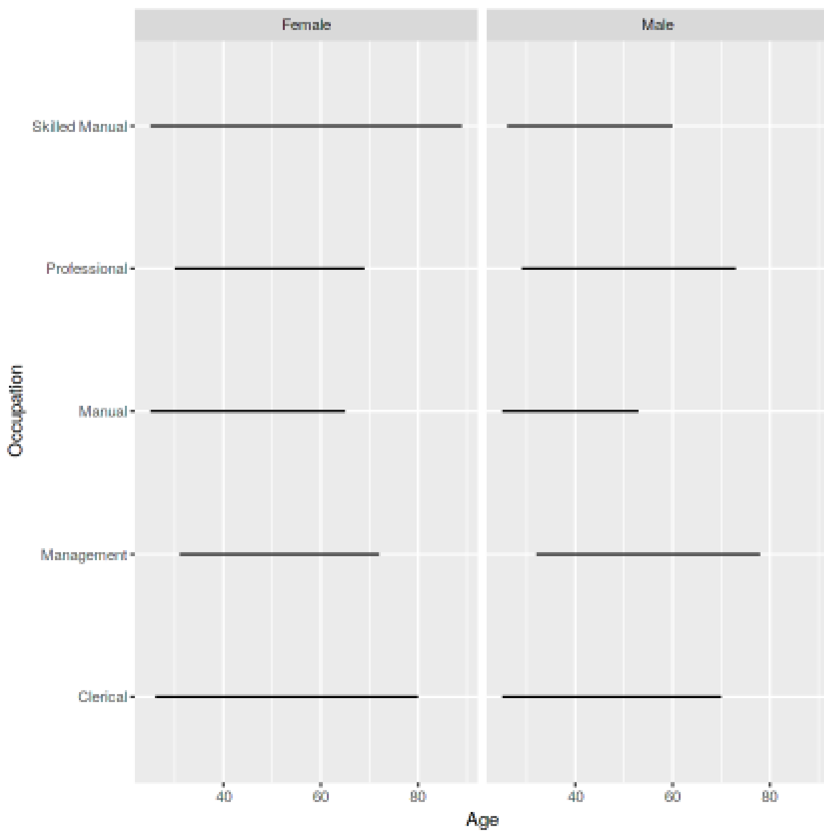
p5 <- ggplot(bike_buyers, aes(x = Age, y = Occupation))

p5 + geom_line(aes(color = Age))



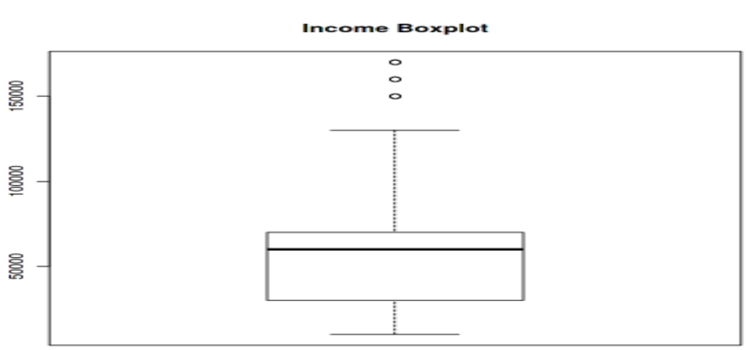
(p5 <- p5 + geom_line() +

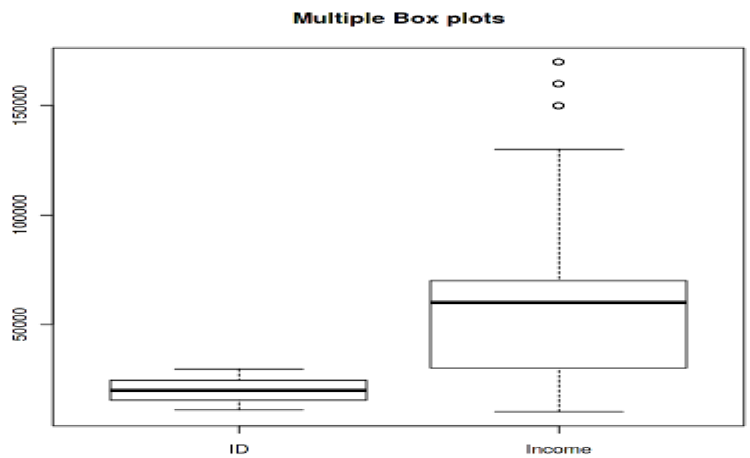
facet_wrap(~Gender, ncol = 10))



boxplot(bike_buyers$Income, main = 'Income Boxplot')

boxplot(bike_buyers[,c(1,4)], main='Multiple Box plots')





OutVals = boxplot(bike_buyers$Income)$out

print(OutVals)

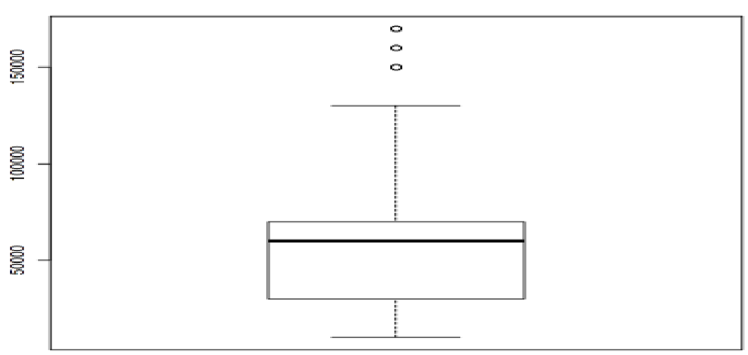
which(bike_buyers$Income %in% OutVals)

x = bike_buyers$Income [!(bike_buyers$Income %in% OutVals)]

boxplot(x)

```
[1] 160000 170000 170000 150000 160000 150000 160000 150000 170000 150000
```

7 · 13 · 44 · 122 · 179 · 260 · 322 · 357 · 830 · 994



EXP-9: Write an R program to illustrate Linear Regression and Multi linear Regression considering suitable CSV file

PROGRAM:

```
# Create sample data

set.seed(123)

size <- 200


# Independent variables

feature1 <- runif(size, 50, 100)

feature2 <- runif(size, 20, 80)

feature3 <- runif(size, 30, 90)


# Dependent variable (for linear and multiple regression)

target <- 2.5 * feature1 + 3 * feature2 + 4 * feature3 + rnorm(size, mean = 0, sd =
10)


# Create data frame

regression_data <- data.frame(Feature1 = feature1, Feature2 = feature2, Feature3
= feature3, Target = target)


# Write to CSV file

write.csv(regression_data, "regression_data.csv", row.names = FALSE)
```

```r
# Load necessary library
install.packages("ggplot2")
library(ggplot2)

# Load data from CSV file
data <- read.csv("regression_data.csv", header = TRUE)

# Split the data into variables
feature1 <- data$Feature1
feature2 <- data$Feature2
feature3 <- data$Feature3
target <- data$Target
```

Linear Regression - Single Predictor

```r
# Simple Linear Regression using Feature1 to predict Target
linear_model <- lm(Target ~ Feature1, data = data)
print(summary(linear_model))
```

 # Plot the Linear Regression Model

```r
ggplot(data, aes(x = Feature1, y = Target)) +
  geom_point(color = "blue") +
  geom_smooth(method = "lm", color = "red", se = FALSE) +
  labs(title = "Linear Regression of Target vs Feature1", x = "Feature1", y = "Target")
```

Multiple Linear Regression

```
# Multiple Linear Regression using Feature1, Feature2, and Feature3 to predict Target
multiple_linear_model <- lm(Target ~ Feature1 + Feature2 + Feature3, data = data)
print(summary(multiple_linear_model))

# Coefficients of the Multiple Linear Regression Model
print(coef(multiple_linear_model))

# Predict Target using the model
predictions <- predict(multiple_linear_model, newdata = data)

# Create a data frame for comparison of actual vs predicted values
comparison <- data.frame(Actual = target, Predicted = predictions)
print(head(comparison))

# Plot actual vs predicted values
ggplot(comparison, aes(x = Actual, y = Predicted)) +
  geom_point(color = "darkgreen") +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  labs(title = "Actual vs Predicted Values (Multiple Linear Regression)",
       x = "Actual Target", y = "Predicted Target")
```

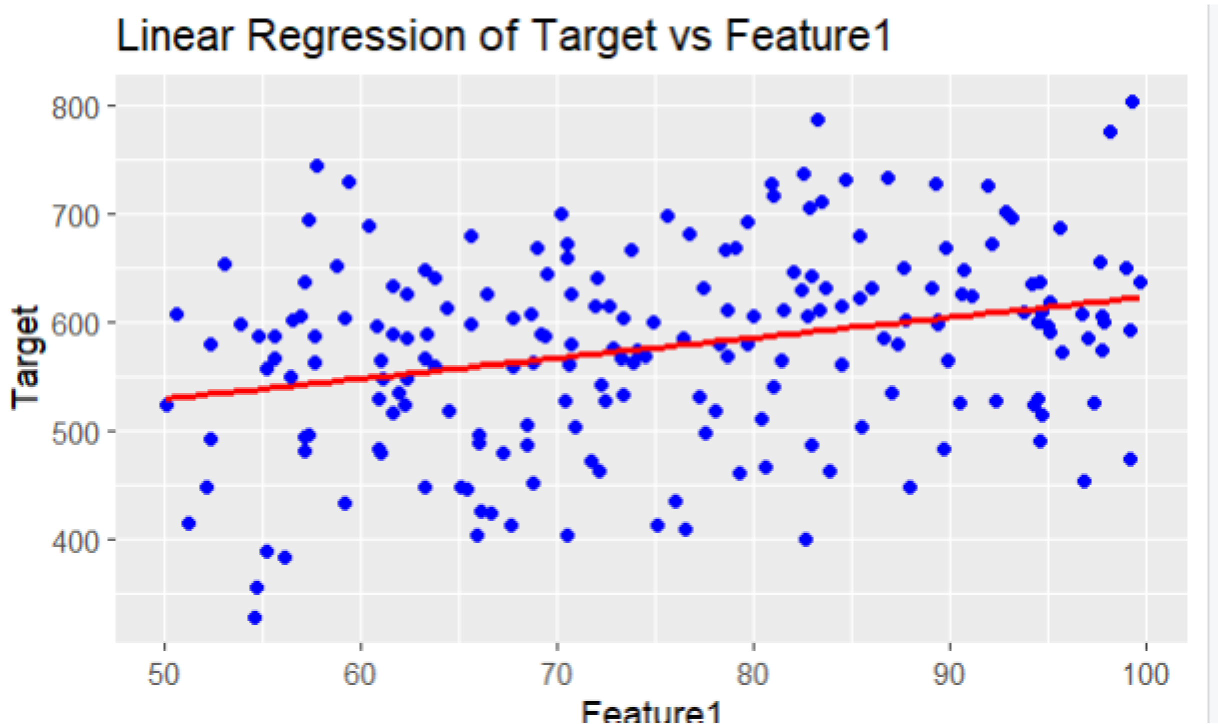
Output:

Linear Model

linear_model	list [12] (S3: lm)	List of length 12
coefficients	double [2]	436.40 1.87
residuals	double [200]	56.65 -1.44 104.14 23.44 -32.24 46.26 ...
effects	double [200]	-8163.2 -361.4 100.3 20.2 -35.4 42.0 ...
rank	integer [1]	2
fitted.values	double [200]	557 604 568 612 618 534 ...
assign	integer [2]	0 1
qr	list [5] (S3: qr)	List of length 5
df.residual	integer [1]	198
xlevels	list [0]	List of length 0

call	language	lm(formula = Target ~ Feature1, data ...
[[1]]	symbol	\`lm\`
formula	language	Target ~ Feature1
data	symbol	\`data\`
terms	formula	Target ~ Feature1
model	list [200 x 2] (S3: data.frame)	A data.frame with 200 rows and 2 col...

Linear Regression:



Comparison:

	Actual	Predicted
1	613.4185	621.4580
2	602.1321	609.4519
3	672.2569	682.4436
4	635.8698	645.3884
5	585.5577	588.6082
6	580.4033	579.5107
7	585.9681	606.2496
8	490.3559	486.9889
9	498.4473	485.9569
10	576.8325	556.6703
11	600.4514	586.1392

Multiple Linear Regression: