PG Department of Data Science

Bishop Heber College (Autonomous),

Tiruchirappalli-620 017, Tamil Nadu

Ranked **34th** at the National Level by MHRD through **NIRF 2023**

(Nationally Reaccredited at the **A++** Grade by NAAC with a CGPA of **3.69 out of 4**)

BONAFIDE CERTIFICATE

Name :_____

Register No : _____

Class : _____

Course Title : **P24DS1P3**

Course Title : **Essential Statistics with R Programming Lab**

Certified that this is the bonafide record of work done by me during Odd Semester of 2024 - 2025 and submitted to the Practical Examination on_____

Staff In-Charge **Head of the Department**

Examiners

1._____

2._____

INDEX

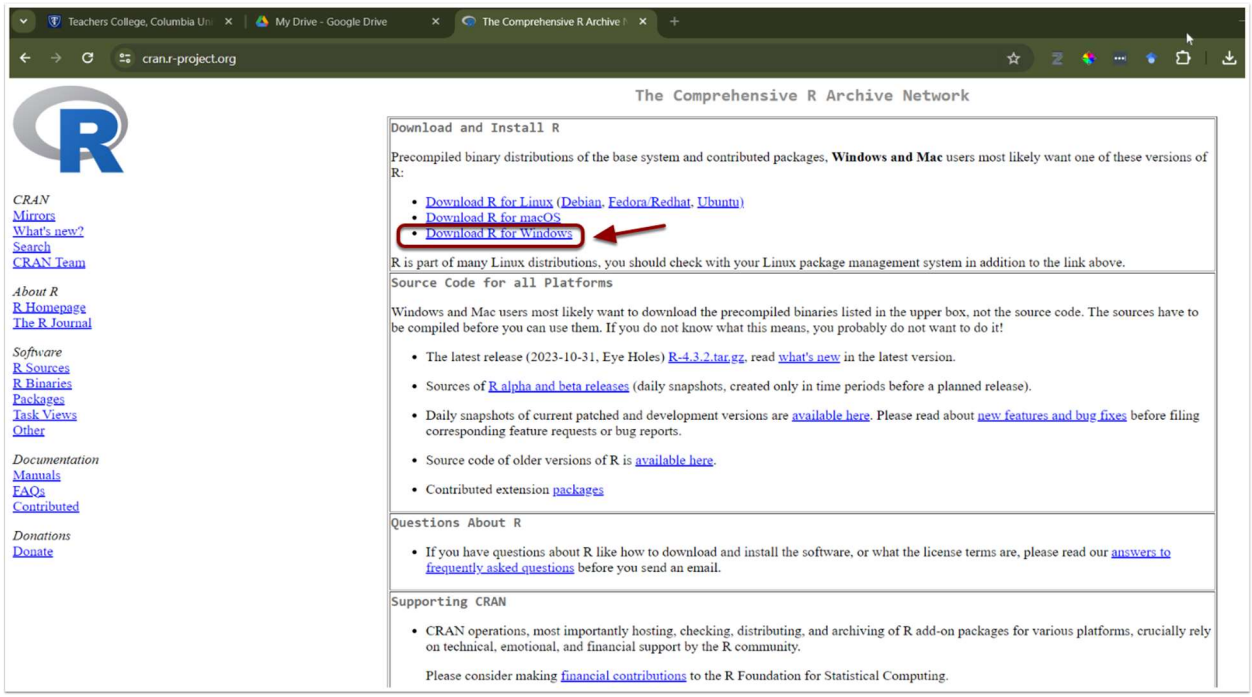
S.No	Date	Title	Page No	Signature
1		The basics of R-Programming		
2		Implement different data structures in R.		
3		Read a CSV file and analyze the data in the file in R.		
4		Find mean, median and standard deviation for a given discrete probability distribution.		
5		Represent a given data in the form of Graphs.		
6		Perform a simple linear regression analysis.		
7		Perform a chi-square test on a contingency table to test for independence.		
8		Perform a one-way ANOVA on a given dataset.		
9		Perform a two-sample t-test to compare means of two independent samples.		
10		Plotting various probability distributions		

Install R and RStudio for Windows

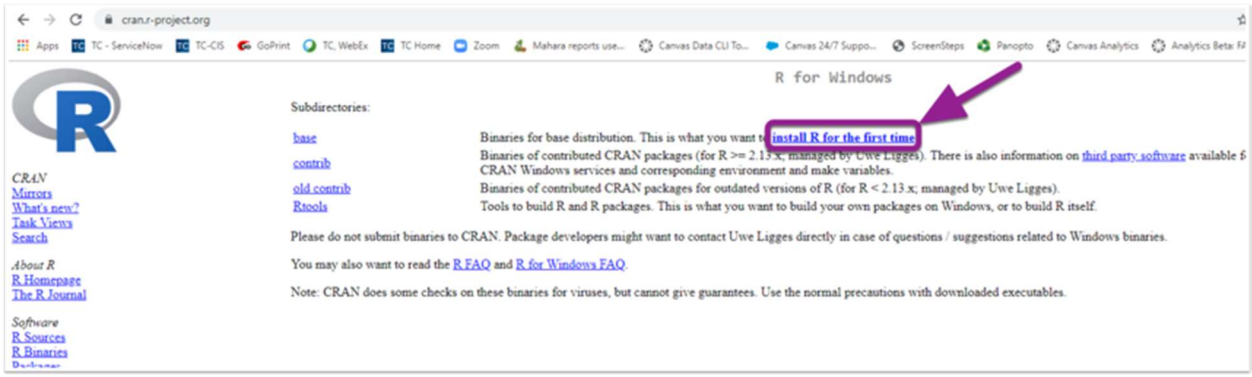
R and RStudio are two different programs that work together. R is the programming language and RStudio is this is a front end program that lets you write R code, view plots, etc. Each program must be downloaded separately.

❖ To install R, go to cran.r-project.org

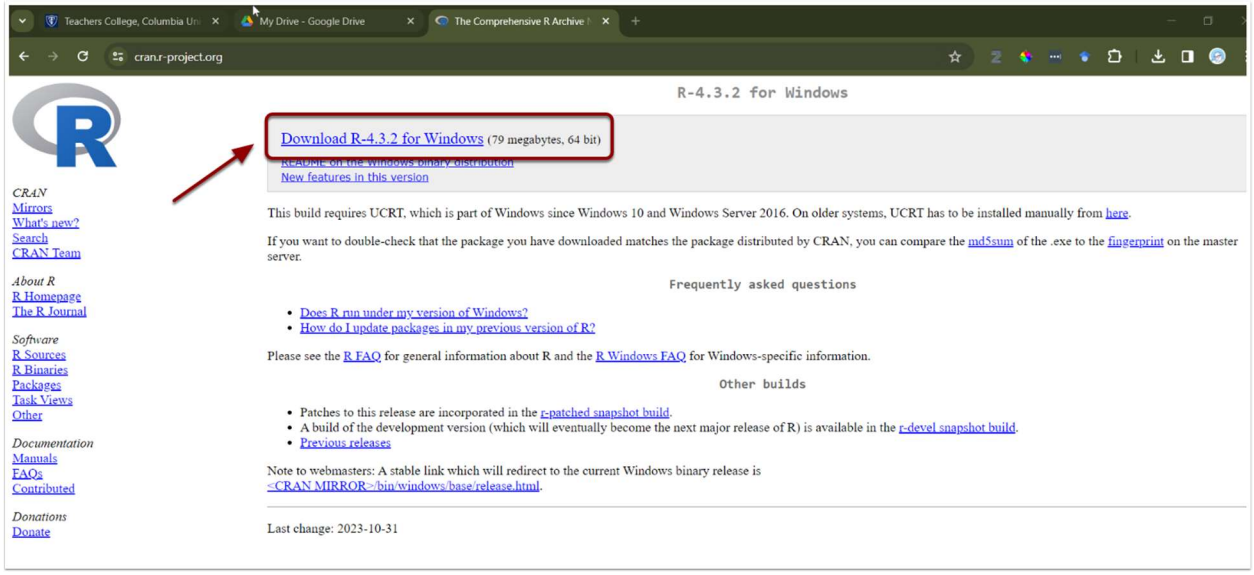
❖ **Click Download R for Windows**



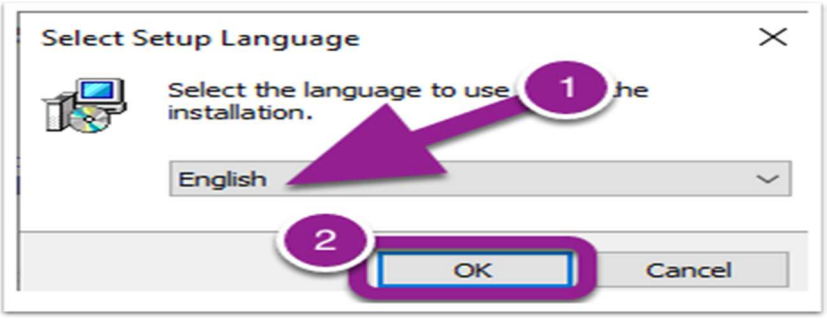
❖ Install R Click on install R for the first time.



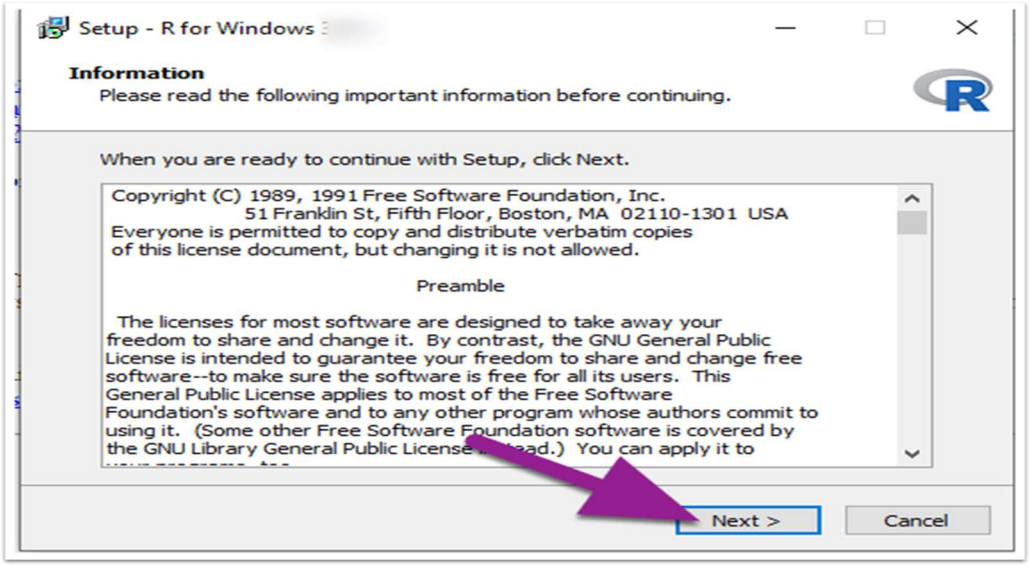
❖ Click Download R for Windows. Open the downloaded file.



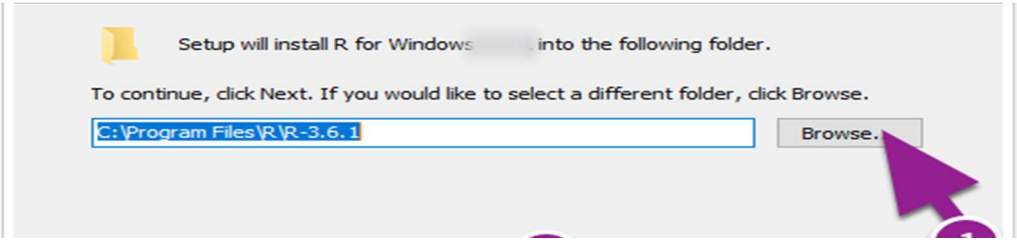
❖ Select the language you would like to use during the installation. Then click OK.



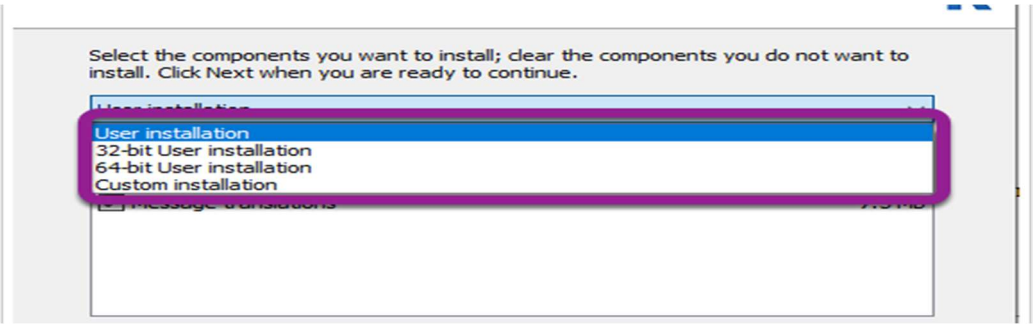
❖ Click Next.



- Select where you would like R to be installed. It will default to your Program Files on your C Drive. Click Next.

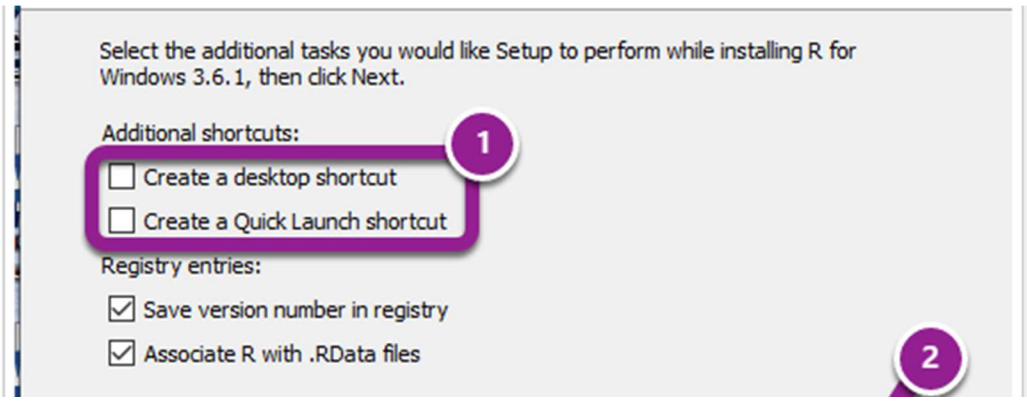


- You can then choose which installation you would like.

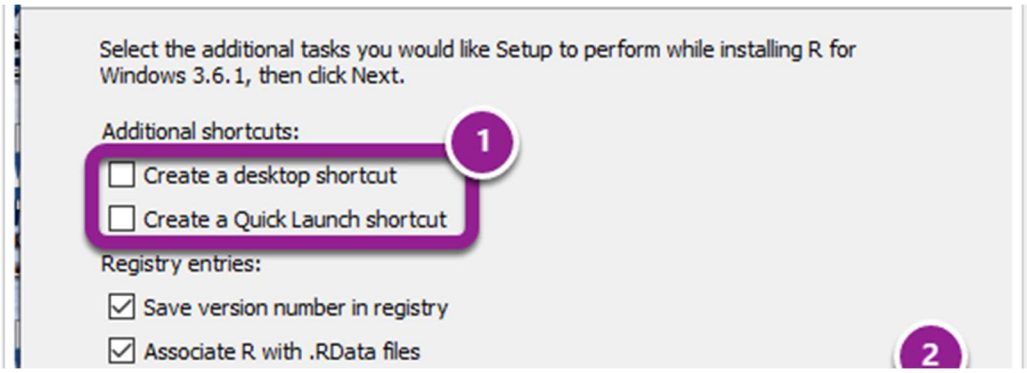


- Then specify if you want to customized your startup or just use the defaults. Then click Next.

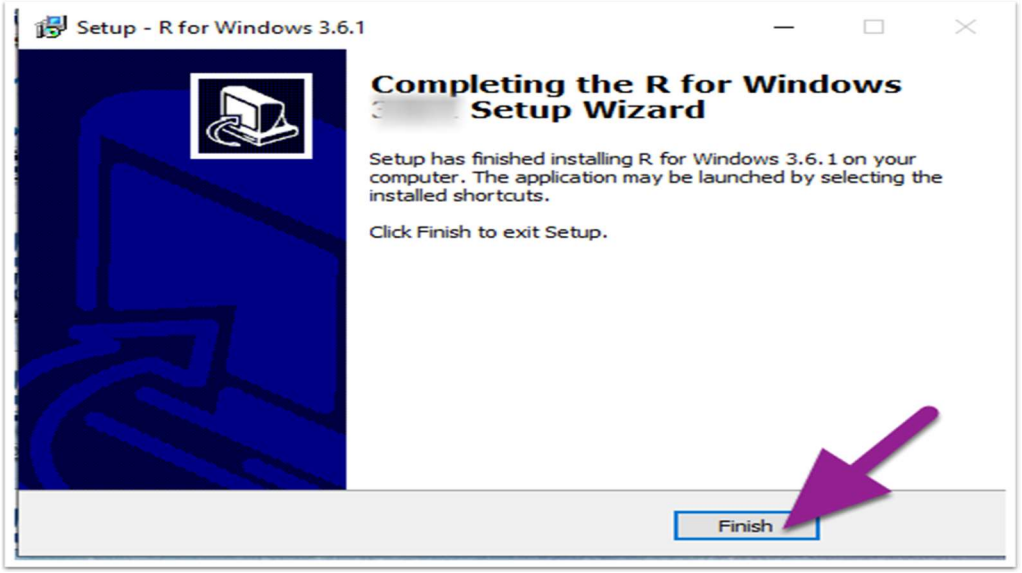
- Then you can choose the folder that you want R to be saved within or the default if the R folder that was created. Once you have finished, click Next.



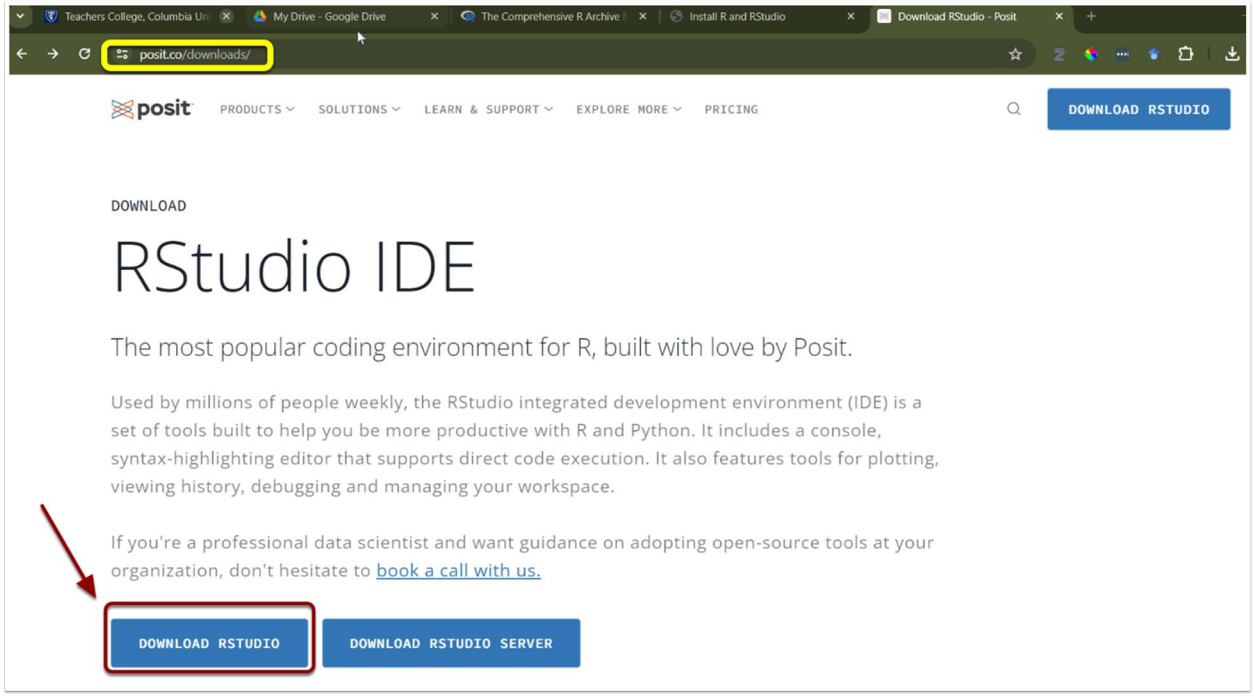
- You can then select additional shortcuts if you would like. Click Next.



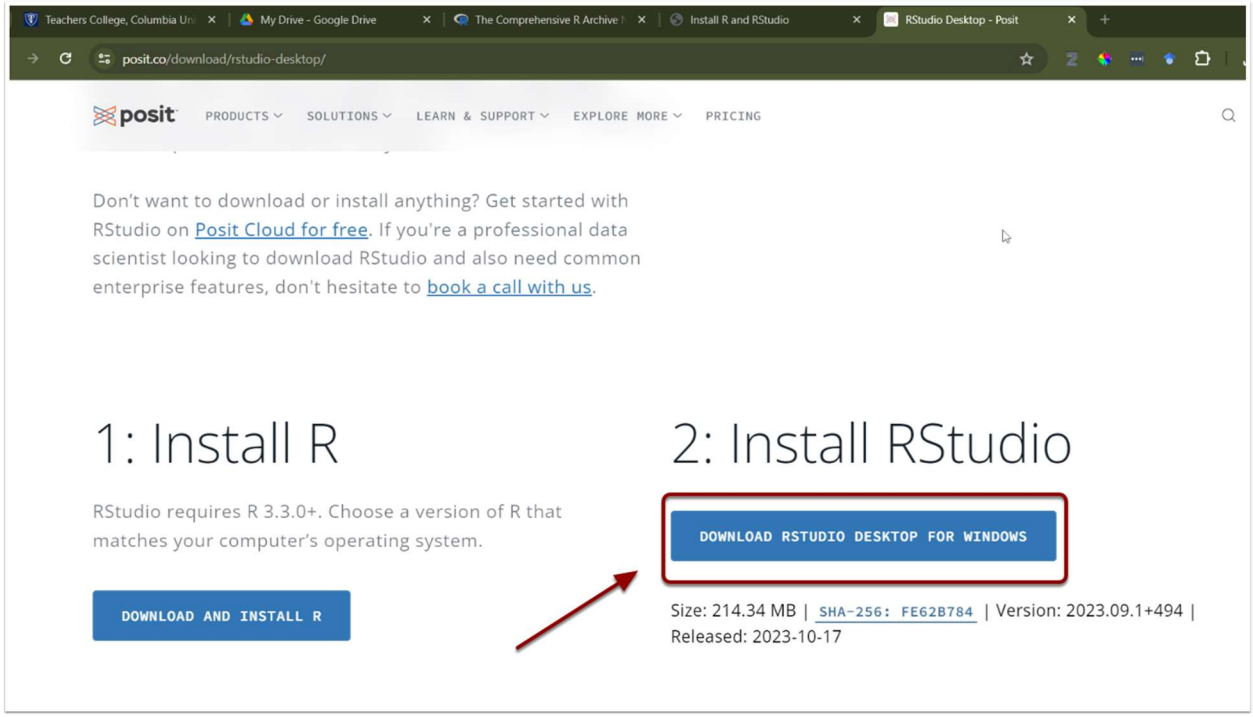
- Click Finish.



❖ Next, download RStudio. Go to https://posit.co/downloads/

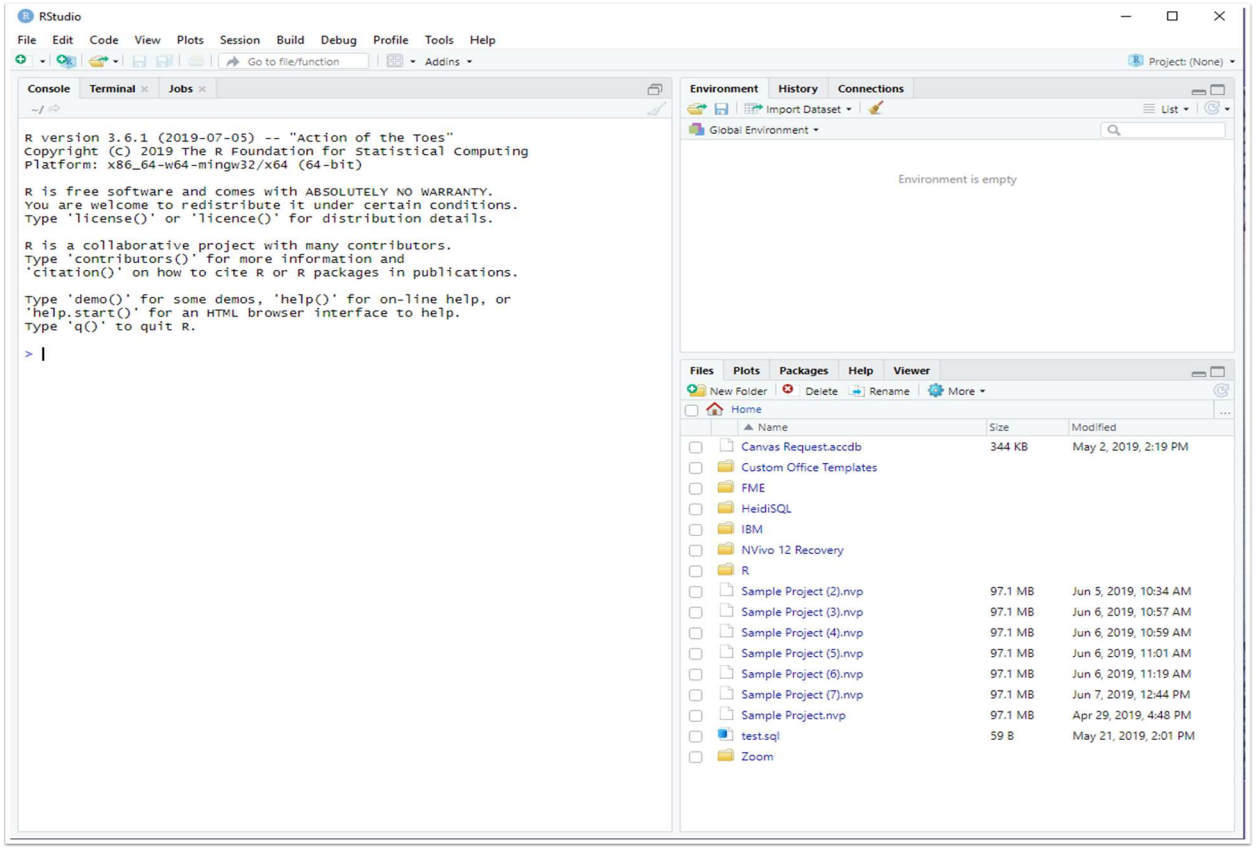


❖ Click Download RStudio.



❖ Once the packet has downloaded, the Welcome to RStudio Setup Wizard will open. Click Next and go through the installation steps.

❖ After the Setup Wizard finishing the installation, RStudio will open.



Exercise 1: The Basics of R Programming

Aim:

To introduce students to the fundamental concepts of R programming, including data types, variables, operators, control statements, and functions.

Procedure:

- **Data Types:** Understand the basic data types in R: numeric, integer, character, and logical.
- **Variables:** Learn how to declare and use variables to store and manipulate data.
- **Operators:** Explore arithmetic, relational, and logical operators.
- **Control Statements:** Practice using control statements such as if, else, for, and while loops.
- **Functions:** Write and use functions to encapsulate reusable code.

Program

```
# 1. Data Types
numeric_var <- 10.5          # Numeric data type
integer_var <- 7L            # Integer data type
char_var <- "R Programming"          # Character data type
logical_var <- TRUE          # Logical data type

# Display Data Types
cat("Data Types:\n")
cat("Numeric Variable:", numeric_var, "Type:", typeof(numeric_var), "\n")
cat("Integer Variable:", integer_var, "Type:", typeof(integer_var), "\n")
cat("Character Variable:", char_var, "Type:", typeof(char_var), "\n")
cat("Logical Variable:", logical_var, "Type:", typeof(logical_var), "\n\n")

# 2. Variables
a <- 5
b <- 3

# Arithmetic Operations
sum <- a + b
difference <- a - b
product <- a * b
quotient <- a / b

cat("Arithmetic Operations:\n")
cat("Sum:", sum, "\n")
cat("Difference:", difference, "\n")
cat("Product:", product, "\n")
cat("Quotient:", quotient, "\n\n")
```

```
# 3. Operators
# Relational Operations
is_equal <- (a == b)
is_greater <- (a > b)

cat("Relational Operations:\n")
cat("Is Equal:", is_equal, "\n")
cat("Is Greater:", is_greater, "\n\n")

# Logical Operations
logical_and <- (a > 2) & (b < 5)
logical_or <- (a < 2) | (b > 1)

cat("Logical Operations:\n")
cat("Logical AND:", logical_and, "\n")
cat("Logical OR:", logical_or, "\n\n")

# 4. Control Statements
cat("Control Statements:\n")

# If-Else Statement
if (a > b) {
  cat("a is greater than b\n")
} else {
  cat("a is not greater than b\n")
}

# For Loop
cat("For Loop:\n")
for (i in 1:3) {
  cat("Iteration:", i, "\n")
}

# 5. Functions
# Define a Function
add_numbers <- function(x, y) {
  return(x + y)
}

# Use the Function
result <- add_numbers(10, 15)
```

```
cat("Function Result:\n")
cat("Sum of 10 and 15 is:", result, "\n")
```

Output:

Data Types:
Numeric Variable: 10.5 Type: double
Integer Variable: 7 Type: integer
Character Variable: R Programming Type: character
Logical Variable: TRUE Type: logical

Arithmetic Operations:
Sum: 8
Difference: 2
Product: 15
Quotient: 1.666667

Relational Operations:
Is Equal: FALSE
Is Greater: TRUE

Logical Operations:
Logical AND: TRUE
Logical OR: TRUE
Control Statements:
a is greater than b

For Loop:
Iteration: 1
Iteration: 2
Iteration: 3

Function Result:
Sum of 10 and 15 is: 25

Result:

The exercise successfully demonstrates the basic concepts of R programming. The output confirms the correct use of data types, variables, operators, control statements, and functions in R, validating the implementation and understanding of these fundamental concepts.

Exercise: 1.1: Write a R Script to calculate whether the given number is Odd or Even

Program

Output:

Exercise: 1.2: Write a R Script to calculate factorial of the given value

Program:

Output:

Exercise 2: Program to implement different data structures in R

Aim:

To introduce students to vectors in R, which are basic data structures for storing data of the same type. The exercise will focus on creating, accessing, and manipulating vectors, as well as performing common operations.

Procedure:

- **Creating Vectors**: Learn how to create vectors using the c() function and other methods.
- **Accessing Elements**: Access elements in a vector using indexing and logical conditions.
- **Vector Operations**: Perform arithmetic and logical operations on vectors.
- **Manipulating Vectors**: Practice adding and removing elements from vectors and modifying existing elements.
- **Vector Functions**: Use built-in vector functions like sum(), length(), and mean() to manipulate vector data.

Program:

```
# 1. Creating Vectors
numeric_vector <- c(1.2, 3.4, 5.6, 7.8)   # Numeric vector
cat("Vector Created:\n")
cat("Numeric Vector:", numeric_vector, "\n")

# 2. Accessing Elements
cat("Accessing Elements:\n")
cat("First element of numeric_vector:", numeric_vector[1], "\n")
cat("Elements greater than 3 in numeric_vector:", numeric_vector[numeric_vector > 3], "\n")
cat("Second to third elements in numeric_vector:", numeric_vector[2:3], "\n\n")

# 3. Vector Operations
# Arithmetic Operations on Numeric Vectors
sum_vector <- numeric_vector + 2
product_vector <- numeric_vector * 2

cat("Vector Operations:\n")
cat("Sum of numeric_vector + 2:", sum_vector, "\n")
cat("Product of numeric_vector * 2:", product_vector, "\n\n")

# 4. Manipulating Vectors
cat("Manipulating Vectors:\n")
# Adding a new element
numeric_vector <- c(numeric_vector, 9.0)
cat("Numeric Vector after adding 9.0:", numeric_vector, "\n")
```

```
# Modifying an element
numeric_vector[2] <- 4.5
cat("Numeric Vector after modifying second element:", numeric_vector, "\n")

# Removing the first element
numeric_vector <- numeric_vector[-1]
cat("Numeric Vector after removing the first element:", numeric_vector, "\n\n")

# 5. Vector Functions
cat("Vector Functions:\n")
cat("Sum of elements in numeric_vector:", sum(numeric_vector), "\n")
cat("Length of numeric_vector:", length(numeric_vector), "\n")
cat("Mean of numeric_vector:", mean(numeric_vector), "\n")
```

Output:

Numeric Vector: 1.2 3.4 5.6 7.8

Accessing Elements:

First element of numeric_vector: 1.2
Elements greater than 3 in numeric_vector: 3.4 5.6 7.8
Second to third elements in numeric_vector: 3.4 5.6

Vector Operations:

Sum of numeric_vector + 2: 3.2 5.4 7.6 9.8
Product of numeric_vector * 2: 2.4 6.8 11.2 15.6

Manipulating Vectors:

Numeric Vector after adding 9.0: 1.2 3.4 5.6 7.8 9
Numeric Vector after modifying second element: 1.2 4.5 5.6 7.8 9
Numeric Vector after removing the first element: 4.5 5.6 7.8 9

Vector Functions:

Sum of elements in numeric_vector: 26.9
Length of numeric_vector: 4
Mean of numeric_vector: 6.725

Result:

The exercise successfully demonstrates how to create, access, and manipulate vector data structures in R. The output confirms the correct application of vector operations, logical operations, and the use of built-in vector functions, providing a solid understanding of vector manipulation in R programming.

Exercise 2.1: Program to implement the Marix Data Structure

Aim:

To introduce students to matrix manipulation in R, focusing on matrix creation, accessing elements, performing operations, and using built-in matrix functions.

Procedure:

1. Create Two Matrices
   - ❖ Create matrix1: 3x3 numeric matrix
   - ❖ Create matrix2: 3x3 numeric matrix
   - ❖ Display both matrix-1 and matrix-2

2. Access the Matrix elements
   - ❖ Access element at (2,3) in matrix 1
   - ❖ Access second row of Matrix 2
   - ❖ Accessing third column of Matrix1

3. Perform the following Matrix Operations
   - ❖ Matrix Addition
   - ❖ Matrix Subtraction
   - ❖ Matrix Multiplication (Element-wise)
   - ❖ Matrix Multiplication (Dot Product)

4. Demonstrate the following Matrix Functions
   - ❖ Transpose of Matrix
   - ❖ Row Sums and Column Sums
   - ❖ Matrix Inverse (using solve function)

Program:

Output

Result:

The exercise demonstrates how to create, access, and manipulate matrices in R. Matrix operations (addition, subtraction, multiplication) and matrix functions (transpose, row sums, column sums) have been successfully applied.

Exercise 2.2: R Program to demonstrate the manipulation of array data structure

Aim:

To introduce students to the array data structure in R and demonstrate how to create, access, and manipulate arrays.

Procedure:

1. Creating Arrays
   - ❖ Create a 3x3x2 array (3 rows, 3 columns, 2 matrices) called array1
   - ❖ Create another 3x3x2 array called array2
   - ❖ Display the arrays

2. Access and Print the elements in an Array
   - ❖ Access an element at position (2,3,1)
   - ❖ Access the second row of first matrix
   - ❖ Access the third column of second matrix

3. Array Operations
   - ❖ Perform Element-wise Addition
   - ❖ Perform Element-wise Subtraction
   - ❖ Perform Element-wise Multiplication

4. Built-in Functions for Arrays
   - ❖ Calculate Sum of all elements in the array
   - ❖ Calculate Row sums for each matrix in Array 1
   - ❖ Calculate Column sums for each matrix in Array 1

Program:

Output

Result:

The exercise successfully demonstrates the creation and manipulation of array data structures in R. The array operations, element access, and built-in functions have been applied correctly, producing the expected output.

Exercise 2.3: R Program to demonstrate the manipulation of Data Frame data structure

Aim:

To introduce students to the manipulation of data frame data structures in R, including creating data frames, accessing data, performing operations, and modifying data frame contents.

Procedure:

1. Creating a Data Frame
   - ❖ Create astudents_df data frame with the columns RollNo,Name and Marks
   - ❖ Display the Data Frame

2. Accessing Data
   - ❖ Access the Name column
   - ❖ Access the second row of the data structure
   - ❖ Access the Marks of 3rd student

3. Modifying Data Frames
   - ❖ Add a new column Grades
   - ❖ Add a new row
   - ❖ Updating the Marks of 2nd student as 80
   - ❖ Display the Data Frame

4. Apply Built-in Functions for Data Frame
   - ❖ Dispaly the Structure of the data frame
   - ❖ Display Summary of the data frame
   - ❖ Display Number of rows and columns ofthe data frame

Program:

Output

Result:

The exercise demonstrates how to create, access, modify, and explore a data frame in R. The output confirms the successful manipulation of data frames, including adding and removing columns, updating values, and using built-in functions to inspect the structure and summary of the data.

Exercise 2.4: R Program to demonstrate the manipulation of List data structure

Aim:

To demonstrate the creation, access, modification, and manipulation of list data structures in R.

Procedure:

1. Creating a List
   - Craete a list nammed as student_list with Name, RollNo and Marks column (Example: Name:Alex RollNo:101, Marks:(60,70,80))
   - Display the List

2. Accessing List Elements
   - Access the Name and RollNo values
   - Access Marks by index

3. Modifying List Elements
   - Modify the Marks as (90, 80, 85)
   - Add a new element Grade as A

4. List Operations
   - Create a new list new_info with the attributes Hobby and GPA(Example:Hobby as "Reading" and GPA as 3.8)
   - Merge both student_list and new_info lists
   - Find the mean of Marks
   - Print the Length of the list

Program:

Output

Result:

The exercise demonstrates the creation, access, modification, and manipulation of lists in R. The output confirms the successful use of various list operations, such as adding, modifying, removing elements, and applying functions.

Exercise 3: Program to Read a CSV File and Analyze the Data in R

Aim:

To introduce students to reading and analyzing CSV files in R. Students will learn how to read a CSV file, explore the structure of the data, summarize the data, and perform basic data analysis.

Procedure:

- **Reading the CSV File**: Understand how to use the read.csv() function to import data from a CSV file into R.
- **Exploring the Data**: Explore the structure and summary of the dataset using functions like str(), summary(), and head().
- **Analyzing the Data**: Perform basic analysis on the dataset, such as calculating the mean, median, and mode of numeric columns, and generating frequency tables for categorical data.

Program

```
# 1. Reading the CSV File "sample_data.csv"
data <- read.csv("sample_data.csv")

# 2. Display the Preview and Summary of the data (first  five records)
cat("Data Preview:\n")
print(head(data,5))

cat("\nSummary of the Data:\n")
print(summary(data))

# 3. Analyzing the Data
# Calculate mean, median, and mode for a numeric column (Assuming 'Age' column exists)
cat("\nBasic Analysis:\n")
age_mean <- mean(data$Age, na.rm = TRUE)
age_median <- median(data$Age, na.rm = TRUE)
age_mode <- as.numeric(names(sort(table(data$Age), decreasing=TRUE)[1]))

cat("Mean Age:", age_mean, "\n")
cat("Median Age:", age_median, "\n")
cat("Mode Age:", age_mode, "\n\n")

# Generate a frequency table for Gender column
gender_table <- table(data$Gender)
cat("Gender Distribution:\n")
print(gender_table)

# Visualize the data using a simple bar plot for 'Gender'
barplot(gender_table, main = "Gender Distribution", col = "lightblue")
```

sample_data.csv:

```
ID,Name,Age,Gender,Salary
1,John,28,Male,50000
2,Emma,32,Female,60000
3,Alex,25,Male,45000
4,Maria,29,Female,52000
5,James,31,Male,55000
6,Sophia,30,Female,58000
7,Chris,26,Male,47000
8,Linda,33,Female,61000
9,David,31,Male,48000
```

Output:

Data Preview:

	ID	Name	Age	Gender	Salary
1	1	John	28	Male	50000
2	2	Emma	32	Female	60000
3	3	Alex	25	Male	45000
4	4	Maria	29	Female	52000
5	5	James	31	Male	55000

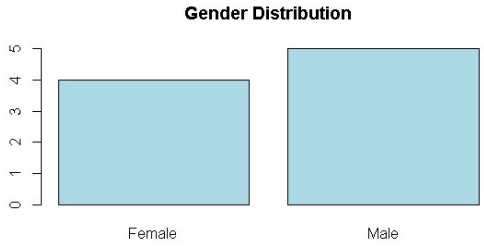
Summary of the Data:

ID	Name	Age	Gender	Salary
Min. :1	Length:9	Min. :25.00	Length:9	Min. :45000
1st Qu.:3	Class :character	1st Qu.:28.00	Class :character	1st Qu.:48000
Median :5	Mode :character	Median :30.00	Mode :character	Median :52000
Mean :5		Mean :29.44		Mean :52889
3rd Qu.:7		3rd Qu.:31.00		3rd Qu.:58000
Max. :9		Max. :33.00		Max. :61000

Basic Analysis:
Mean Age: 29.44444
Median Age: 30
Mode Age: 31

Gender Distribution:

Female	Male
4	5



Result:

The program successfully demonstrates how to read a CSV file and perform basic data exploration and analysis in R. The output provides the insights such as the mean, median, mode of numeric columns, and frequency tables for categorical data.

Exercise 3.1: Perform the Data Analysis for the given Employee Data Set

Employee.csv

Id,Name,Age,Gender,Department,Salary
1,John Smith,28,Male,IT,75000
2,Emily Johnson,34,Female,HR,68000
3,Michael Brown,29,Male,Finance,72000
4,Sarah Davis,31,Female,Marketing,64000
5,David Wilson,45,Male,IT,90000
6,Linda Taylor,38,Female,HR,70000
7,Robert Lee,26,Male,Finance,61000
8,Susan Harris,30,Female,Marketing,67000
9,James Clark,41,Male,IT,85000
10,Mary Allen,27,Female,Finance,62000

Procedure

- Reading the CSV File
- Display the preview of the data (first five records)
- Display the summary of the data
- Calculate mean, median, and mode for a SALARY column
- Display the calculated mean, median and mode values
- Generate a frequency table for a SALARY Column and Display it
- Visualize the data using a simple bar plot for SALARY

Program:

Output:

Exercise 4: Program to Find Mean, Median, and Standard Deviation for a Given Discrete Probability Distribution

Aim:

To write a program in R that calculates the mean, median, and standard deviation for a given discrete probability distribution.

Procedure:

1. **Define the Discrete Probability Distribution**: Provide the set of discrete values and their corresponding probabilities.
2. **Calculate the Mean**: Use the formula mean = Σ(x * P(x)), where x is the value and P(x) is the corresponding probability.
3. **Calculate the Median**: Use the median of the distribution by finding the middle value after sorting.
4. **Calculate the Standard Deviation**: Compute the standard deviation using σ = √Σ((x - mean)^2 * P(x)).
5. **Execute the Program** and verify the calculated results.

Program:

```
# Step 1: Define the discrete probability distribution
values <- c(1, 2, 3, 4, 5)      # Given values
probabilities <- c(0.1, 0.2, 0.3, 0.2, 0.2)  # Corresponding probabilities

# Step 2: Calculate the Mean
mean_value <- sum(values * probabilities)

# Step 3: Calculate the Median
median_value <- median(rep(values, probabilities * 100))

# Step 4: Calculate the Standard Deviation
variance <- sum((values - mean_value)^2 * probabilities)
std_dev <- sqrt(variance)

# Output the results
cat("Mean:", mean_value, "\n")
cat("Median:", median_value, "\n")
cat("Standard Deviation:", std_dev, "\n")
```

Output:

Mean: 3.3
Median: 3
Standard Deviation: 1.356466

Result:

The solution is implemented, and the mean, median, and standard deviation are correctly calculated. The output is verified.

Exercise 5: Program to Represent a Given Data in the Form of Graphs

Aim:

To write a program in R that visualizes given data using different types of graphs, such as bar charts, histograms, and scatter plots.

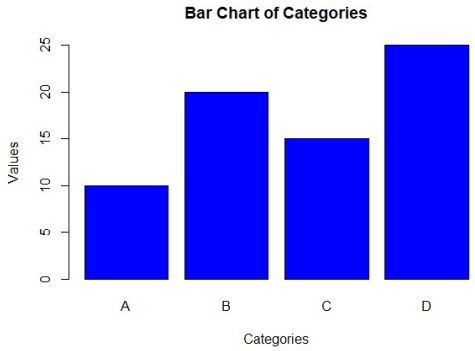
Procedure:

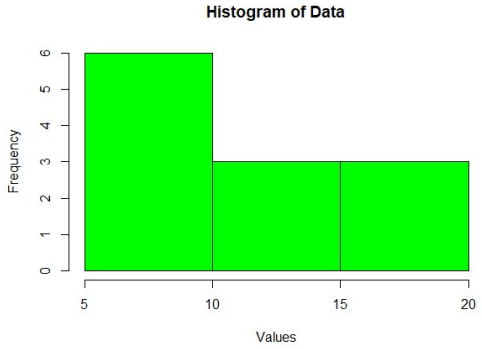
1. **Input Data**: Provide the dataset to be visualized.
2. **Create a Bar Chart**: Use the barplot() function to represent categorical data.
3. **Create a Histogram**: Use the hist() function to show the distribution of continuous data.
4. **Create a Scatter Plot**: Use the plot() function to show the relationship between two numeric variables.
5. **Execute the Program** and verify that the graphs are correctly displayed.

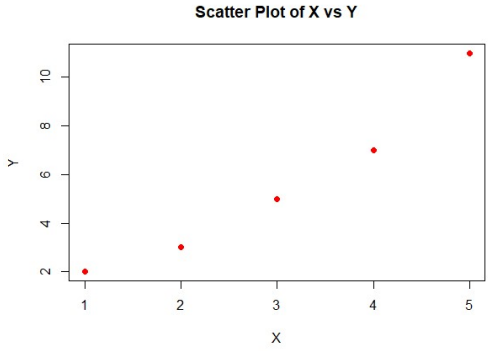
Program:

```
# Step 1: Define the data
# For Bar Chart
categories <- c("A", "B", "C", "D")
values <- c(10, 20, 15, 25)

# For Histogram
data <- c(5, 7, 8, 9, 10, 10, 12, 14, 15, 16, 17, 20)

# For Scatter Plot
x <- c(1, 2, 3, 4, 5)
y <- c(2, 3, 5, 7, 11)

# Step 2: Create a Bar Chart
barplot(values, names.arg = categories, main = "Bar Chart of Categories", xlab = "Categories", ylab = "Values", col = "blue")

# Step 3: Create a Histogram
hist(data, breaks = 5, main = "Histogram of Data", xlab = "Values", ylab = "Frequency", col = "green")

# Step 4: Create a Scatter Plot
plot(x, y, main = "Scatter Plot of X vs Y", xlab = "X", ylab = "Y", pch = 19, col = "red")
```

Output:







Result:

The solution is implemented, and the data is accurately represented in the form of graphs. The output is verified.

Exercise 5.1: Program to Represent a Given Data in the Form of Graphs

Aim:

To write a program in R that visualizes given data using different types of graphs, such as bar charts, histograms, and scatter plots.

Procedure:

- **Histogram**: Learn to create a histogram to visualize the distribution of numerical data.
- **Bar Plot**: Create a bar plot to compare categorical data.
- **Line Chart**: Create a line chart to visualize trends over time or an ordered sequence.
- **Pie Chart**: Represent data as a pie chart for proportional visualization.
- **Box Plot**: Create a box plot to display the distribution based on summary statistics.
- **Scatter Plot**: Visualize the relationship between two variables using a scatter plot.

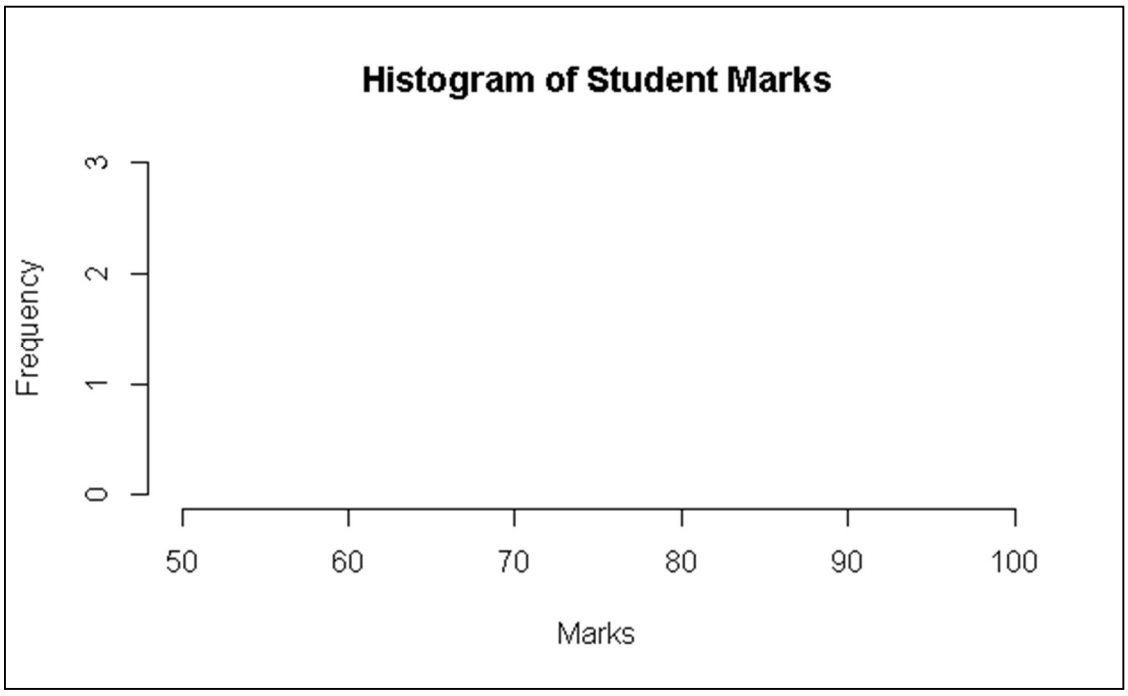
Program:

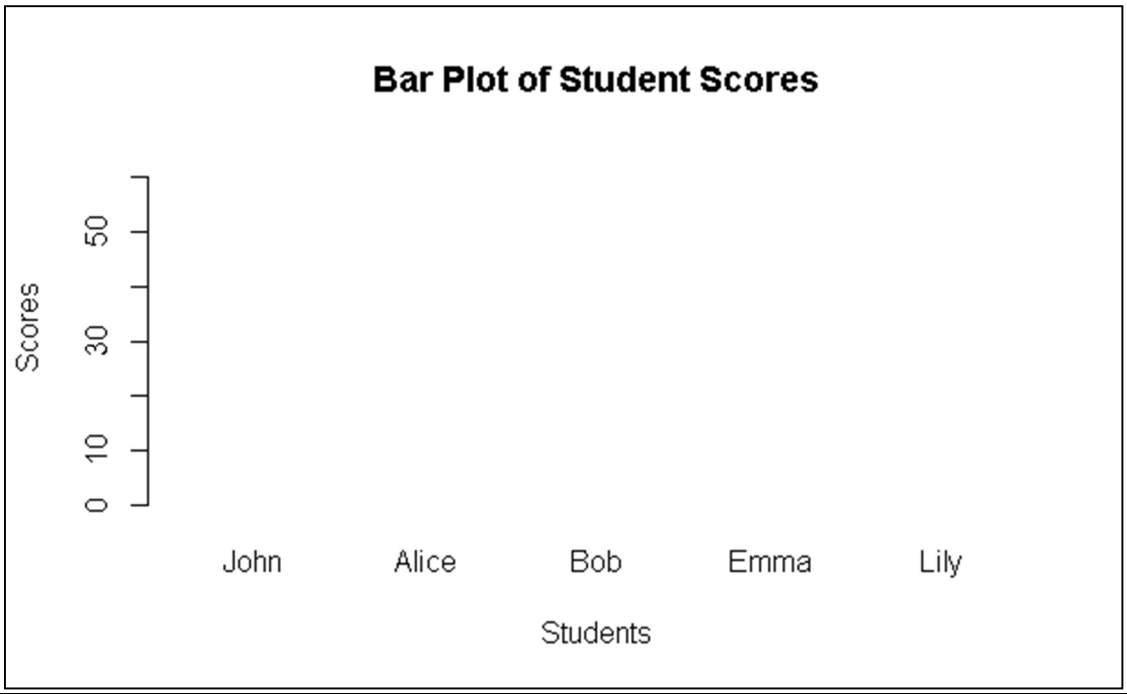
```
# 1. Histogram
# Creating a dataset of student marks
student_marks <- c(55, 65, 70, 72, 73, 90, 95, 100, 75)

# Plotting a Histogram
hist(student_marks,
   main = "Histogram of Student Marks",
   xlab = "Marks",
   ylab = "Frequency",
   col = "lightblue",
   border = "black")

# 2. Bar Plot
# Creating a dataset of student names and marks
student_names <- c("John", "Alice", "Bob", "Emma", "Lily")
student_scores <- c(45, 55, 65, 55, 35)

# Plotting a Bar Plot
barplot(student_scores,
     names.arg = student_names,
     main = "Bar Plot of Student Scores",
     xlab = "Students",
     ylab = "Scores",
     col = "lightgreen",
     border = "black")
```

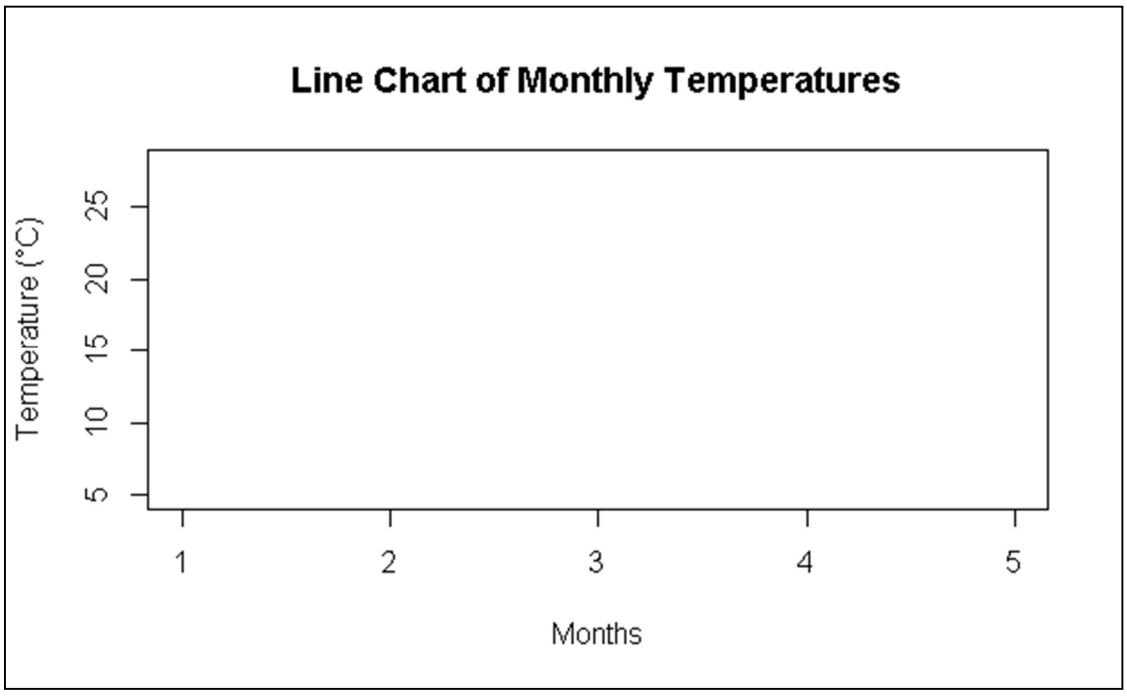
```
# 3. Line Chart
# Creating a dataset of monthly temperatures
months<-c(1,2,3,4,5)
temperature <- c(10, 15, 5, 20, 28)

# Plotting a Line Chart
plot(months, temperature,
    type = "o",
    col = "blue",
    main = "Line Chart of Monthly Temperatures",
    xlab = "Months",
    ylab = "Temperature (°C)")

# 4. Pie Chart
# Creating a dataset of department-wise students
dept <- c("IT", "Math", "Physics", "Biology")
students <- c(120, 80, 60, 40)

# Plotting a Pie Chart
pie(students,
   labels = dept,
   main = "Pie Chart of Department-Wise Students",
   col = rainbow(length(dept)))

# 5. Box Plot
# Creating a dataset of exam scores
exam_scores <- c(55, 65, 70, 80, 85, 90, 95, 100, 50, 75)

# Plotting a Box Plot
boxplot(exam_scores,
      main = "Box Plot of Exam Scores",
      ylab = "Scores",
      col = "lightcoral",
      border = "black")

# 6. Scatter Plot
# Creating datasets for height and weight
height <- c(150, 160, 165, 170, 175)
weight <- c(50, 55, 60, 65, 70)

# Plotting a Scatter Plot
plot(height, weight,
```

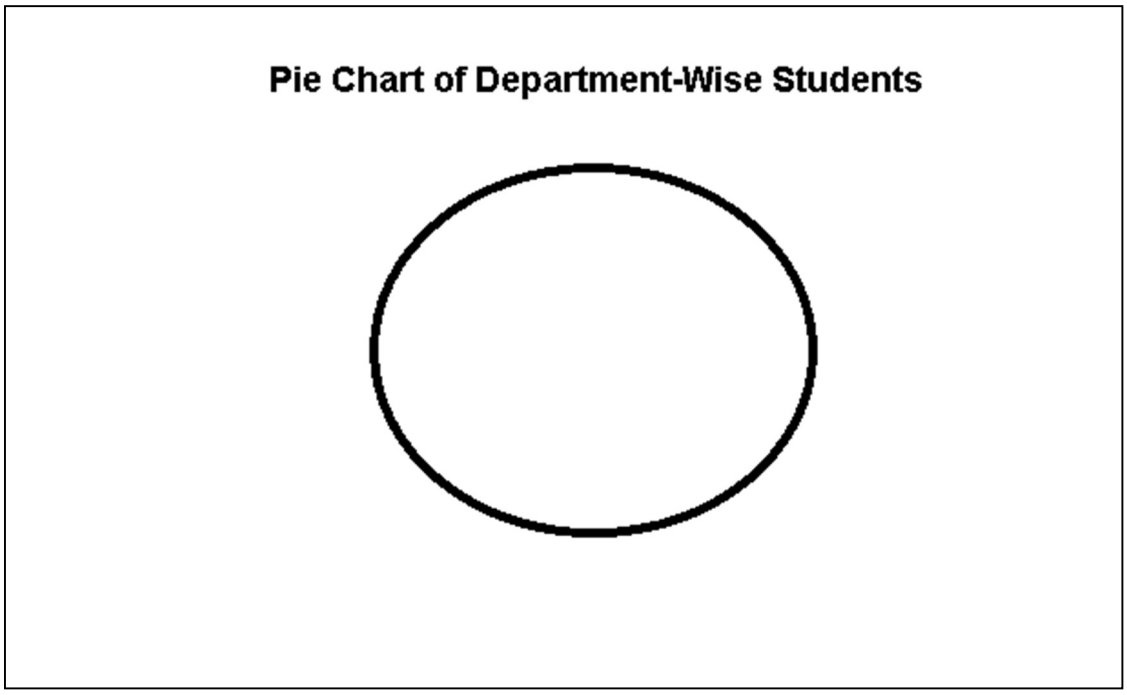
```
  main = "Scatter Plot of Height vs. Weight",
  xlab = "Height (cm)",
  ylab = "Weight (kg)",
  pch = 16,
  col = "darkred")
```

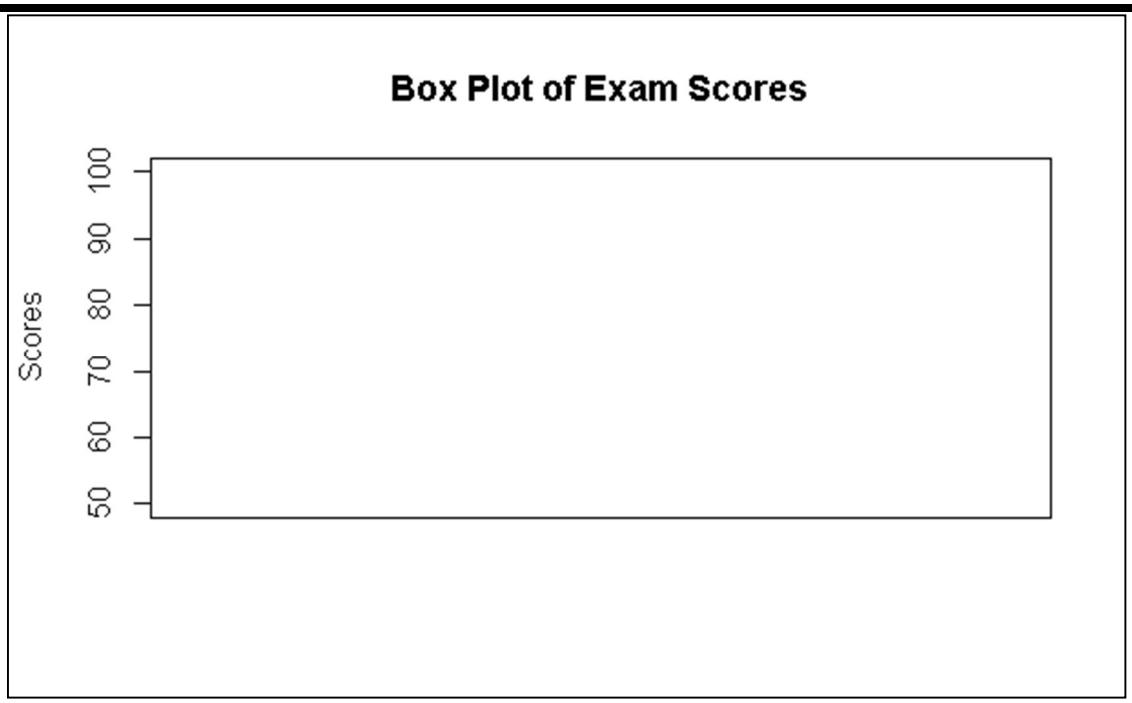
Output:

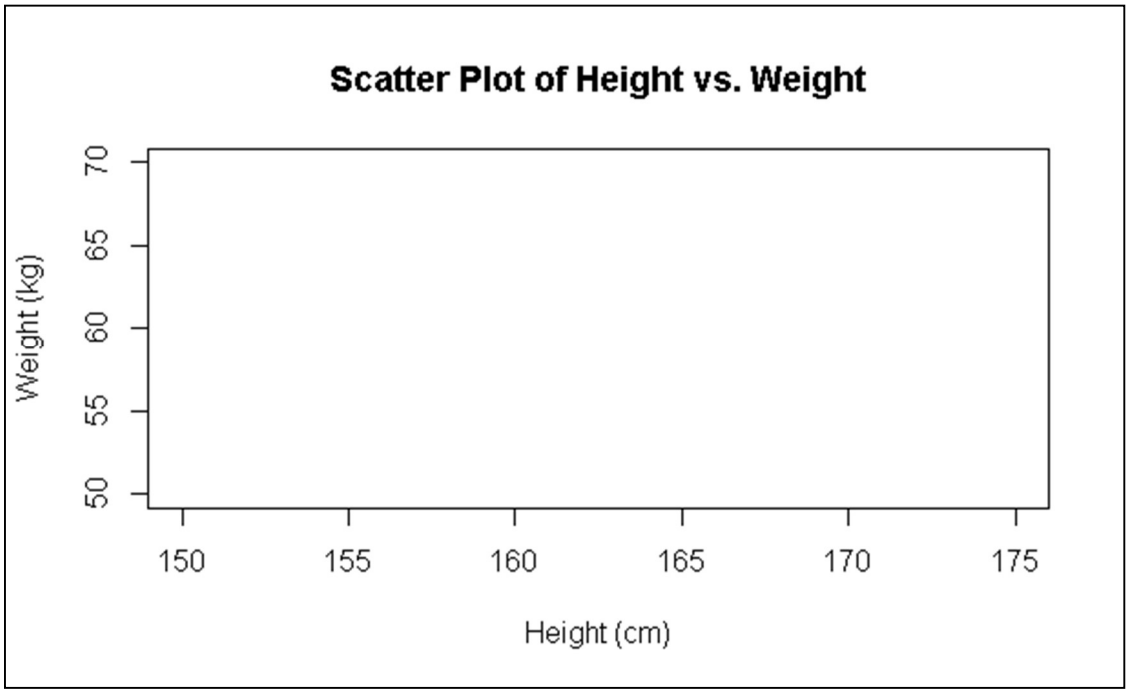





Line Chart of Monthly Temperatures
Temperature (°C)
5
10
15
20
25
1
2
3
4
5
Months


Pie Chart of Department-Wise Students





Result:

This exercise demonstrates the creation of various types of charts in R, including a histogram, bar plot, line chart, pie chart, box plot, and scatter plot, successfully visualizing different types of data.

Exercise 6: Simple Linear Regression Analysis in R

Aim:

To perform a simple linear regression analysis in R, predicting the dependent variable based on the independent variable.

Procedure:

- **Create Dataset**: Use a dataset with two variables: one dependent and one independent.
- **Fit Linear Model**: Fit a linear model using the lm() function.
- **View Model Summary**: Analyze the summary of the regression model.
- **Plot Regression Line**: Visualize the regression line on a scatter plot.

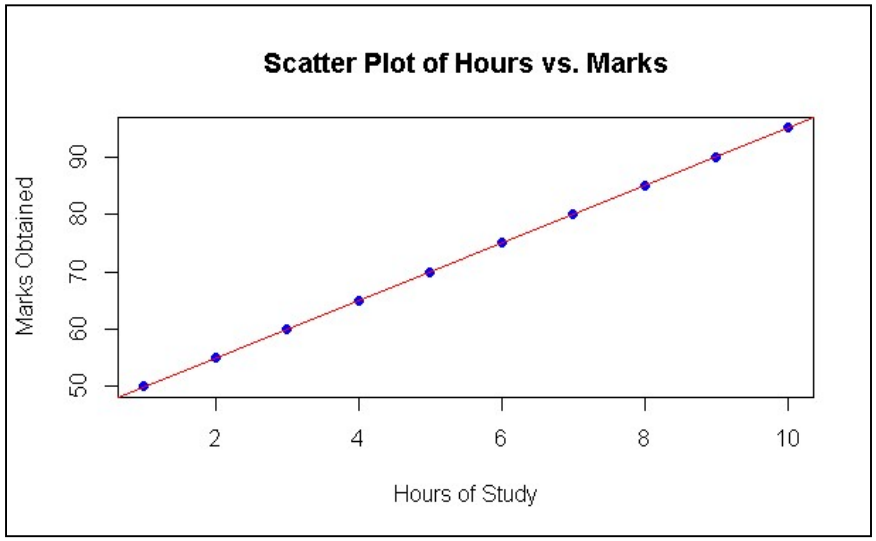
Program:

```
# 1. Create Dataset
# Independent variable: Hours of study
# Dependent variable: Marks obtained
hours <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
marks <- c(50, 55, 60, 65, 70, 75, 80, 85, 90, 95)

# Plot the scatter plot
plot(hours, marks,
   main = "Scatter Plot of Hours vs. Marks",
   xlab = "Hours of Study",
   ylab = "Marks Obtained",
   pch = 16,
   col = "blue")

# 2. Perform Simple Linear Regression
model <- lm(marks ~ hours)

# 3. View Model Summary
print(summary(model))

# 4. Plot the Regression Line
abline(model, col = "red")

# 5. Predict Marks for 6.5 hours of study
predicted_marks <- predict(model, data.frame(hours = 6.5))
cat("Predicted Marks for 6.5 hours of study:", predicted_marks, "\n")
```

Output:

```
Call:
lm(formula = marks ~ hours)

Residuals:
     Min       1Q   Median       3Q      Max
-4.155e-15 -1.582e-15  9.575e-16  1.032e-15  4.688e-15

Coefficients:
             Estimate Std. Error   t value Pr(>|t|)
(Intercept) 4.500e+01  1.848e-15 2.434e+16   <2e-16 ***
hours       5.000e+00  2.979e-16 1.678e+16   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.706e-15 on 8 degrees of freedom
Multiple R-squared:      1,      Adjusted R-squared:      1
F-statistic: 2.817e+32 on 1 and 8 DF,  p-value: < 2.2e-16
```

Predicted Marks for 6.5 hours of study: 77.5



Result:

The exercise demonstrates how to fit a simple linear regression model, interpret the results, and make predictions based on the model.

Exercise 6.1: Simple Linear Regression Analysis in R (Height vs. Weight)

Aim:

To perform a simple linear regression analysis in R, predicting weight based on height.

Procedure:

- **Create Dataset:** Use a dataset with two variables: height (independent variable) and weight (dependent variable).
- **Fit Linear Model:** Fit a linear model using the lm() function.
- **View Model Summary:** Analyze the summary of the regression model to understand the relationship between height and weight.
- **Plot Regression Line:** Visualize the regression line on a scatter plot.
- Predict Weight for a height of 172.5 cm

Program:

Output:

Result:

The exercise demonstrates how to fit a simple linear regression model, interpret the results, and make predictions based on the model.

Exercise 7: Chi-Square Test for Independence in R

Aim:

To perform a chi-square test on a contingency table to determine if two categorical variables are independent.

Procedure:

- **Create a Contingency Table**: Create a table showing the frequency of occurrences of two categorical variables.
- **Perform Chi-Square Test**: Use the chisq.test() function to perform the test.
- **Analyze the Result**: Interpret the test result by analyzing the p-value and the test statistic.

Program

```
# 1. Create Contingency Table
# Let's consider a study about the preference of two types of sports (Football and Basketball)
# among males and females in a sample of 100 individuals.

# Contingency Table: Rows = Gender, Columns = Sport Preference
gender_sport <- matrix(c(30, 10, 20, 40), nrow = 2, byrow = TRUE,
                       dimnames = list("Gender" = c("Male", "Female"),
                                       "Sport" = c("Football", "Basketball")))

# Display the contingency table
cat("Contingency Table:\n")
print(gender_sport)

# 2. Perform Chi-Square Test for Independence
chi_square_test <- chisq.test(gender_sport)

# 3. View Test Results
cat("\nChi-Square Test Results:\n")
print(chi_square_test)

# 4. Interpretation
if (chi_square_test$p.value < 0.05) {
  cat("\nConclusion: The variables are not independent (reject the null hypothesis).\n")
} else {
  cat("\nConclusion: The variables are independent (fail to reject the null hypothesis).\n")
}
```

Output:

Contingency Table:

	Sport	
Gender	Football	Basketball
Male	30	10
Female	20	40

Chi-Square Test Results:

Pearson's Chi-squared test with Yates' continuity correction

data: gender_sport
X-squared = 15.042, df = 1, p-value = 0.0001052

Conclusion: The variables are not independent (reject the null hypothesis).

Result:

The chi-square test for independence has been performed, and the result shows whether there is a significant association between gender and sport preference.

Exercise 7.1: Chi-Square Test for Independence in R

Aim:

To perform a chi-square test on a contingency table to determine if two categorical variables are independent.

Procedure:

- Create Contingency Table: A survey of preference for different types of cuisine (Italian, Chinese) by age group (Young, Middle-aged)

	Cuisine Preference	
Age Group	Italian	Chinese
Young	25	30
Middle-aged	20	35

- Display the contingency table
- Perform Chi-Square Test for Independence
- View the Chi-Square Test Results
- Interpret the result

Programe:

Output:

Result:

The chi-square test for independence has been performed, and the result shows whether there is a significant association between gender and sport preference.

Exercise 8: One-Way ANOVA in R

Aim:

To perform a one-way ANOVA on a given dataset to test if there are statistically significant differences between the means of multiple groups.

Procedure:

- **Create a Dataset**: Define the dataset with multiple groups.
- **Perform One-Way ANOVA**: Use the aov() function to conduct the analysis.
- **Interpret the Results**: Analyze the p-value and F-statistic to determine whether the group means are significantly different.

Code:

```
# 1. Create a Dataset
# Example: Test scores of students from three different groups (Group A, Group B, and Group C)
scores <- c(88, 90, 85, 92, 95, 78, 82, 87, 91, 86, 83, 89, 94, 80, 77)
groups <- factor(c(rep("Group A", 5), rep("Group B", 5), rep("Group C", 5)))

# Combine into a data frame
data <- data.frame(scores, groups)

# Display the dataset
cat("Dataset:\n")
print(data)

# 2. Perform One-Way ANOVA
anova_result <- aov(scores ~ groups, data = data)

# 3. View the ANOVA Summary
cat("\nOne-Way ANOVA Results:\n")
print(summary(anova_result))

# 4. Interpretation
# Check p-value from the ANOVA summary
p_value <- summary(anova_result)[[1]]["Pr(>F)"][1]
if (p_value < 0.05) {
  cat("\nConclusion: There is a significant difference between the group means (reject the null hypothesis).\n")
} else {
  cat("\nConclusion: There is no significant difference between the group means (fail to reject the null hypothesis).\n")
}
```

Output:

Dataset:

	scores	groups
1	88	Group A
2	90	Group A
3	85	Group A
4	92	Group A
5	95	Group A
6	78	Group B
7	82	Group B
8	87	Group B
9	91	Group B
10	86	Group B
11	83	Group C
12	89	Group C
13	94	Group C
14	80	Group C
15	77	Group C

One-Way ANOVA Results:

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
groups	2	93.7	46.87	1.625	0.237
Residuals	12	346.0	28.83		

Conclusion: There is no significant difference between the group means (fail to reject the null hypothesis).

Result:

This program demonstrates how to perform a one-way ANOVA in R, allowing you to analyze whether the means of different groups are statistically different.

Exercise 8.1: One-Way ANOVA in R Using Weight and Diets

Aim:

To perform a one-way ANOVA on a dataset comparing the average weight of individuals on three different diets.

Procedure:

- **Create a Dataset:** Example: Weights of individuals on three different diets (Diet A, Diet B, and Diet C)
  weights <- c(68, 72, 65, 70, 74, 60, 63, 67, 69, 64, 76, 78, 71, 73, 75)
  diets <- factor(c(rep("Diet A", 5), rep("Diet B", 5), rep("Diet C", 5)))
- Combine into a data frame
- Display the dataset
- Perform One-Way ANOVA
- View the ANOVA Summary
- Interpret the result

Program:

Output

Result:

This program demonstrates how to perform a one-way ANOVA in R, allowing you to analyze whether the means of different groups are statistically different.

Exercise 9: Two-Sample T-Test in R

Aim:

To perform a two-sample t-test to compare the means of two independent samples.

Procedure:

- **Create a Dataset**: Define two independent samples (e.g., weights of individuals from two different groups).
- **Perform Two-Sample T-Test**: Use the t.test() function to compare the means of the two samples.
- **Interpret the Results**: Analyze the p-value and t-statistic to determine whether the means are significantly different.

Program:

```
# 1. Create a Dataset
# Example: Weights of individuals in Group 1 and Group 2
group1 <- c(68, 72, 65, 70, 74)  # Weights in Group 1
group2 <- c(60, 63, 67, 69, 64)  # Weights in Group 2

# Display the samples
cat("Group 1 Weights:\n")
print(group1)
cat("Group 2 Weights:\n")
print(group2)

# 2. Perform Two-Sample T-Test
t_test_result <- t.test(group1, group2)

# 3. View the T-Test Results
cat("\nTwo-Sample T-Test Results:\n")
print(t_test_result)

# 4. Interpretation
# Check p-value from the t-test result
p_value <- t_test_result$p.value
if (p_value < 0.05) {
  cat("\nConclusion: There is a significant difference between the means of the two groups (reject the null hypothesis).\n")
} else {
  cat("\nConclusion: There is no significant difference between the means of the two groups (fail to reject the null hypothesis).\n")
}
```

Output:

```
Group 1 Weights:
[1] 68 72 65 70 74
Group 2 Weights:
[1] 60 63 67 69 64

Two-Sample T-Test Results:
        Welch Two Sample t-test

data:  group1 and group2
t = 2.3491, df = 7.9999, p-value = 0.04675
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.09542744 10.30457256
sample estimates:
mean of x mean of y
    69.8      64.6
```

Conclusion: There is a significant difference between the means of the two groups (reject the null hypothesis).

Result:

This program demonstrates how to perform a two-sample t-test in R to compare the means of two independent samples. The analysis reveals whether the means of the two groups are significantly different based on the given data.

Exercise 9.1: Two-Sample t-Test in R

Aim:

To perform a two-sample t-test on a dataset comparing the means of two independent samples.

Procedure:

- **Create a Dataset**: Define the dataset with scores from two independent groups.
- **Perform the Two-Sample t-Test**: Use the t.test() function to conduct the test.
- **Interpret the Results**: Analyze the p-value to determine if there is a significant difference between the two groups.

Program:

Output:

Result:

This program demonstrates how to perform a two-sample t-test in R to compare the means of two independent samples. The analysis reveals whether the means of the two groups are significantly different based on the given data.

Exercise 10: Plotting Various Probability Distributions in R

Aim:

To plot different types of probability distributions using R, including Normal, Binomial, Poisson, and Exponential distributions.

Procedure:

- **Create Data for Distributions**: Generate data for different probability distributions.
- **Plot Distributions**: Use the plot(), hist(), and curve() functions to visualize the distributions.
- **Interpret the Graphs**: Analyze the shape and behavior of each distribution.

Program:

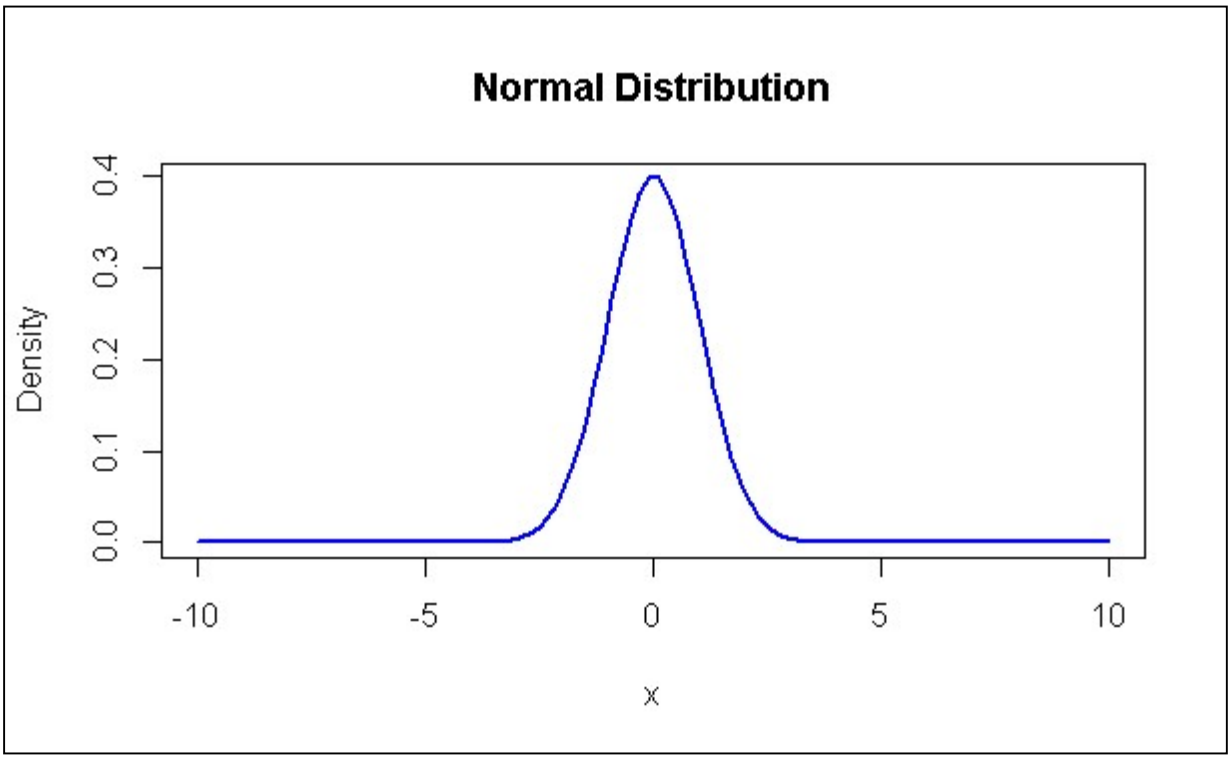
```
# 1. Plot Normal Distribution
cat("Plotting Normal Distribution:\n")
x_normal <- seq(-10, 10, length = 100)
y_normal <- dnorm(x_normal, mean = 0, sd = 1)

# Plot Normal Distribution
plot(x_normal, y_normal, type = "l", col = "blue", lwd = 2,
   main = "Normal Distribution", xlab = "x", ylab = "Density")

# 2. Plot Binomial Distribution
cat("Plotting Binomial Distribution:\n")
x_binom <- 0:10
y_binom <- dbinom(x_binom, size = 10, prob = 0.5)

# Plot Binomial Distribution
barplot(y_binom, names.arg = x_binom, col = "green",
     main = "Binomial Distribution", xlab = "Number of Successes", ylab = "Probability")

# 3. Plot Poisson Distribution
cat("Plotting Poisson Distribution:\n")
x_pois <- 0:15
y_pois <- dpois(x_pois, lambda = 4)

# Plot Poisson Distribution
barplot(y_pois, names.arg = x_pois, col = "red",
     main = "Poisson Distribution", xlab = "Number of Events", ylab = "Probability")

# 4. Plot Exponential Distribution
cat("Plotting Exponential Distribution:\n")
x_exp <- seq(0, 5, length = 100)
```

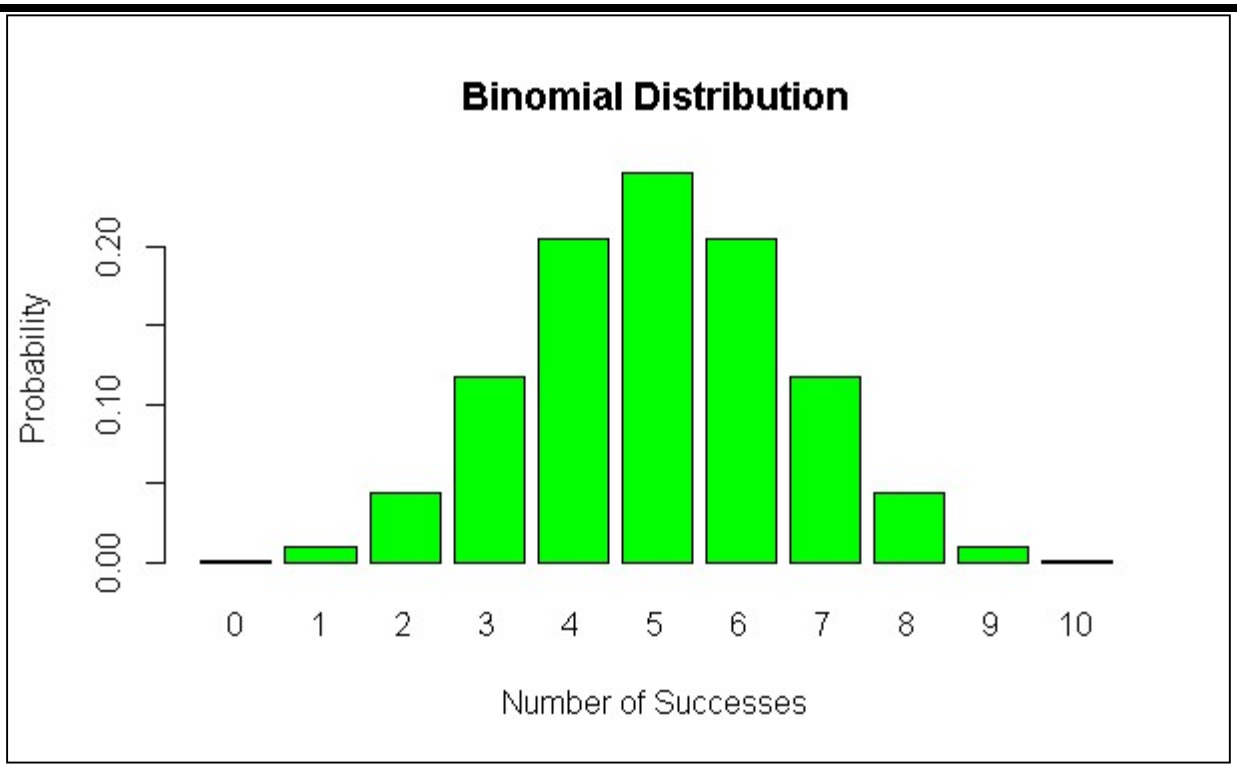
```
y_exp <- dexp(x_exp, rate = 1)

# Plot Exponential Distribution
plot(x_exp, y_exp, type = "l", col = "purple", lwd = 2,
   main = "Exponential Distribution", xlab = "x", ylab = "Density")
```

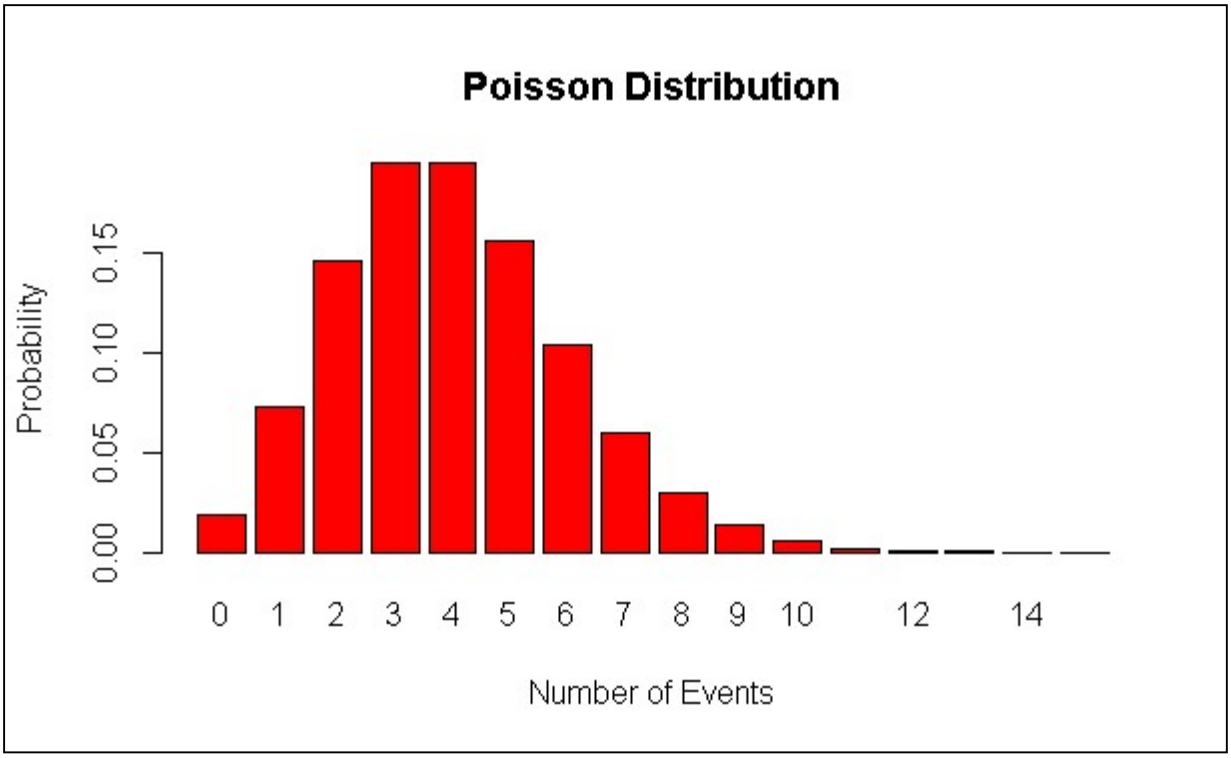
Explanation of Distributions:

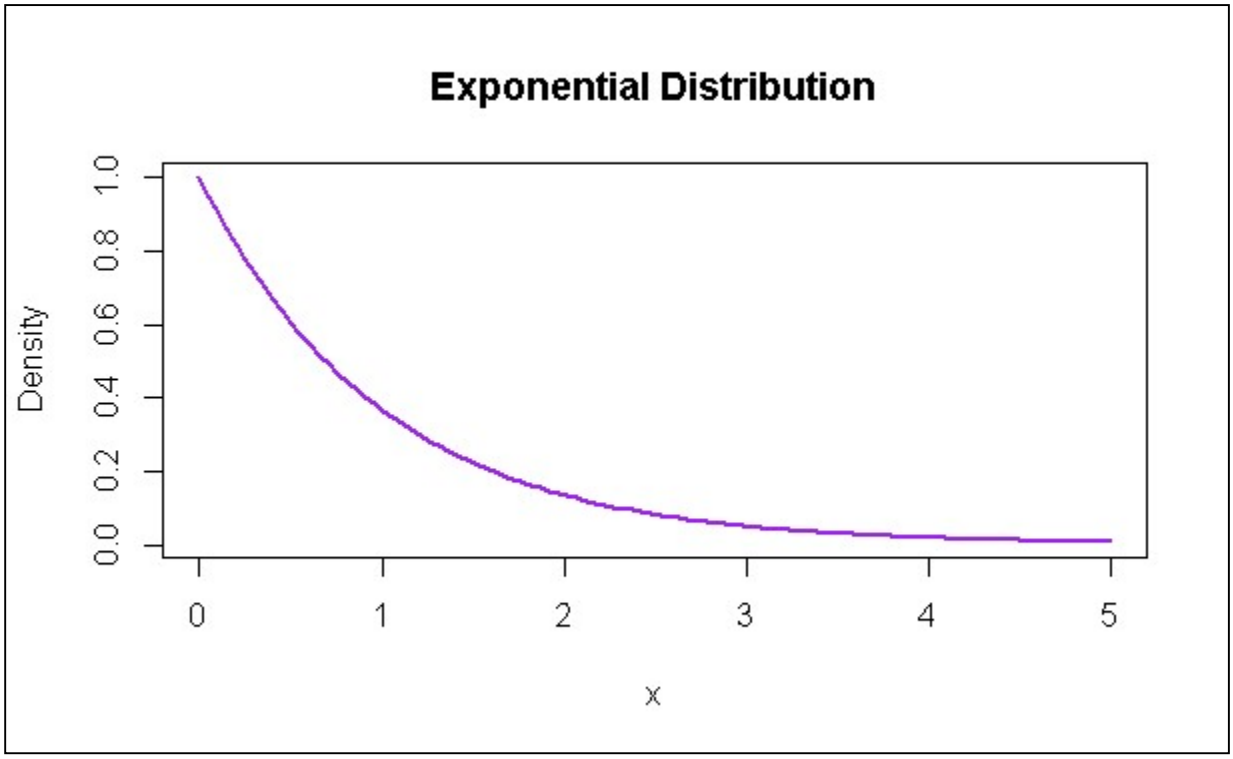
1. **Normal Distribution**:
   - The normal distribution is bell-shaped, symmetric, and describes continuous data. The plot uses the dnorm() function.
2. **Binomial Distribution**:
   - The binomial distribution represents the probability of a given number of successes in a fixed number of independent trials. The plot uses the dbinom() function.
3. **Poisson Distribution**:
   - The Poisson distribution represents the probability of a given number of events happening in a fixed interval of time or space. The plot uses the dpois() function.
4. **Exponential Distribution**:
   - The exponential distribution describes the time between events in a Poisson process. The plot uses the dexp() function.

Output:




Binomial Distribution
Probability
0.00
0.10
0.20
0 1 2 3 4 5 6 7 8 9 10
Number of Successes


Poisson Distribution
Probability
0.00
0.05
0.10
0.15
0 1 2 3 4 5 6 7 8 9 10 12 14
Number of Events



Result:

This program successfully plots various probability distributions, demonstrating the characteristics and behavior of each distribution type.

Exercise 10.1: Plotting Gamma Distribution in R

Aim:

To demonstrate the plotting of a Gamma distribution, which is used to model the time until the occurrence of an event, for multiple shape parameters.

Procedure:

- **Create Data for Gamma Distribution**: Generate data using the dgamma() function for different shape parameters.
- **Plot Gamma Distribution**: Visualize the distribution using the plot() function.

Program:

Output:

Result:

This program successfully demonstrates how to plot the Gamma distribution for different shape parameters, showing how the shape affects the spread and peak of the distribution.