Index

Experiment number	Description	Page No.
Exp 1	R as Calculator Application a. Using with and without R objects on console b. Using mathematical functions on console c. Write an R script, to create R objects for calculator application and save in a specified location in disk.	
Exp 2	Descriptive Statistics in R a. Write an R script to find basic descriptive statistics using summary, str, quartile function on mtcars& cars datasets. b. Write an R script to find subset of dataset by using subset (), aggregate () functions on iris dataset.	
Exp 3	Reading and Writing Different Types of Datasets a. Reading different types of data sets (.txt, .csv) from Web and disk and writing in file in specific disk location. b. Reading Excel data sheet in R. c. Reading XML dataset in R.	
Exp 4	Visualizations a. Find the data distributions using box and scatter plot. b. Find the outliers using plot. c. Plot the histogram, bar chart and pie chart on sample data.	
Exp 5	Correlation and Covariance a. Find the correlation matrix. b. Plot the correlation plot on dataset and visualize giving an overview of relationships among data on iris data. c. Analysis of covariance: variance (ANOVA), if data have categorical variables on iris data	
Exp 6	Regression Model Import a data from web storage. Name the dataset and now do Logistic Regression to find out relation between variables that are affecting the admission of a student in an institute based on his or her GRE score, GPA	

	obtained and rank of the student. Also check the model is fit or not. Require (foreign), require (MASS).	
Exp 7	Multiple Regression Model Apply multiple regressions, if data have a continuous Independent variable. Apply on above dataset.	
Exp 8	Regression Model for Prediction Apply regression Model techniques to predict the data on above dataset.	
Exp 9	Classification Model a. Install relevant package for classification. b. Choose classifier for classification problem. c. Evaluate the performance of classifier.	
Exp 10	Clustering Model a. Clustering algorithms for unsupervised classification. b. Plot the cluster data using R visualizations.	

Experiment 1

Objective:

Create a calculator application in R that allows:

1. Simple arithmetic operations with and without using R objects in the console.
2. Usage of mathematical functions in the console.
3. Creation of an R script to define R objects for calculator functions and save them to a specified location.

Theory

R can function as a calculator by entering arithmetic expressions directly in the console. R evaluates these expressions using operator precedence, meaning certain operations (like multiplication and exponentiation) are computed before others (like addition and subtraction) unless parentheses are used to specify order. Complex expressions can be handled accurately by using parentheses, ensuring that operations occur in the desired sequence. Furthermore, R can store values in objects, which can then be manipulated in subsequent calculations.

R also includes a variety of built-in functions for common mathematical operations (e.g., sqrt, log, sin, etc.), allowing users to perform more complex computations. Additionally, users can create custom functions and scripts, which can be saved to automate repetitive calculations.

a. Using R as a Calculator (Without Objects)

R supports basic arithmetic with operators like + (addition), - (subtraction), * (multiplication), / (division), and ^ (exponentiation).

Examples:

- 4 + 8 → 12
- 5 * 14 → 70
- 7 / 4 → 1.75
- 4 + 5 + 3 → 12
- 4 ^ 3 → 64

Operator Precedence: R evaluates multiplication and exponentiation before addition and subtraction, unless parentheses specify otherwise.

- 4 + 5 * 3 → 19 (evaluates 5 * 3 first)
- (4 + 5) * 3 → 18 (evaluates 4 + 5 first)

Fractional Powers (Roots):

- 4 ^ 0.5 → 2 (square root)
- 16 ^ (1/4) → 2 (fourth root)

Using parentheses allows control over evaluation order, ensuring calculations are performed as intended.

Using R Objects

In R, objects can store values for reuse in calculations. For example, assign 5 to x:

```
x <- 5

x + 2  # Returns 7
```

Here, x remains 5 because we didn't reassign the result. To update x:

```
x <- x + 2

x  # Now x is 7
```

In R, operations like these store results in objects, which can be used for further calculations.

b. Using Mathematical Functions in R

In addition to basic arithmetic, R provides built-in functions for common mathematical operations:

- abs(x) – Absolute value (e.g., abs(-3) → 3)
- sqrt(x) – Square root (e.g., sqrt(16) → 4)
- x ^ y – Exponentiation (e.g., 3 ^ 3 → 27)
- exp(x) – Exponential function (e.g., exp(1) → 2.718)
- log(x) – Natural logarithm (e.g., log(10) → 2.302)
- log10(x) – Logarithm base 10 (e.g., log10(100) → 2)
- sin(x), cos(x), tan(x) – Trigonometric functions (angle in radians, e.g., sin(pi / 2) → 1)
- round(x, n) – Rounds x to n decimal places (e.g., round(pi, 2) → 3.14)
- floor(x) – Rounds down (e.g., floor(14.7) → 14)
- ceiling(x) – Rounds up (e.g., ceiling(14.7) → 15)

These functions enable efficient computation of various mathematical expressions in R.

c. **R Script to Create a Calculator Application**

This R script defines functions for basic arithmetic operations (addition, subtraction, multiplication, and division) and prompts the user to select an operation and input numbers. The result is printed based on the selected operation.

```
# Define functions for basic arithmetic operations

add <- function(x, y) {

  return(x + y)

}

subtract <- function(x, y) {

  return(x - y)

}

multiply <- function(x, y) {

  return(x * y)

}
```

```
divide <- function(x, y) {
  return(x / y)
}
# Prompt user for operation
print("Select operation.")
print("1. Add")
print("2. Subtract")
print("3. Multiply")
print("4. Divide")
# Read user choice and numbers
choice <- as.integer(readline(prompt = "Enter choice [1/2/3/4]:
"))
num1 <- as.integer(readline(prompt = "Enter first number: "))
num2 <- as.integer(readline(prompt = "Enter second number: "))
# Perform selected operation
operator <- switch(choice, "+", "-", "*", "/")
result <- switch(choice, add(num1, num2), subtract(num1, num2),
multiply(num1, num2), divide(num1, num2))
# Display result
print(paste(num1, operator, num2, "=", result))
```

Output:

```
Select operation.
1. Add
2. Subtract
3. Multiply
4. Divide
Enter choice [1/2/3/4]: 1
Enter first number: 8
Enter second number: 3
[1] "8 + 3 = 11"
```

Experiment 2

Objective:

Perform Descriptive Statistics in R.

1. Write an R script for basic descriptive statistics using summary, str, and quantile on mtcars and cars datasets.
2. Write an R script to find subsets of the iris dataset using subset() and aggregate() functions.

Theory and Technique:

Descriptive analysis summarizes data using simple statistics and visualizations, providing insight into the dataset's structure and variability.

1. **Basic Descriptive Statistics in R**:
   Using the mtcars dataset, we can load the data directly (as it's built-in) and use functions like summary(), str(), and quantile() for basic data descriptions.
   - **summary()**: Provides statistical summaries for each variable.
   - **str()**: Displays the internal structure, listing variable types and values.
   - **quantile()**: Calculates quartiles, giving a quick sense of data spread.
2. **Subsetting and Aggregation on iris dataset**:
   The subset() function filters data based on conditions or specific variables, while aggregate() groups data and provides summary statistics for each group.

Code:

Part a: Basic Descriptive Statistics

```
# Load the mtcars dataset

data(mtcars)

myData <- head(mtcars)
```

```
# Summary of a single variable (number of cylinders)
summary_cyl <- summary(myData$cyl)
print("Summary of 'cyl':")
print(summary_cyl)

# Summary of the entire dataset
summary_data <- summary(myData)
print("Summary of 'mtcars' dataset:")
print(summary_data)

# Structure of the dataset
print("Structure of 'mtcars' dataset:")
str(myData)

# Quartiles of miles per gallon (mpg)
quartiles_mpg <- quantile(myData$mpg)
print("Quartiles of 'mpg':")
print(quartiles_mpg)
```

Output:

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    4       6       6       6       6       8

      mpg             cyl             disp             hp             drat
 Min.   :18.10   Min.   :4   Min.   :108.0   Min.   : 93.0   Min.   :2.760
 1st Qu.:19.27   1st Qu.:6   1st Qu.:160.0   1st Qu.:106.2   1st Qu.:3.098
 Median :21.00   Median :6   Median :192.5   Median :110.0   Median :3.500
 Mean   :20.50   Mean   :6   Mean   :211.8   Mean   :117.2   Mean   :3.440
 3rd Qu.:21.30   3rd Qu.:6   3rd Qu.:249.8   3rd Qu.:110.0   3rd Qu.:3.888
 Max.   :22.80   Max.   :8   Max.   :360.0   Max.   :175.0   Max.   :3.900
       wt             qsec             vs              am              gear
 Min.   :2.320   Min.   :16.46   Min.   :0.0   Min.   :0.0   Min.   :3.0
 1st Qu.:2.684   1st Qu.:17.02   1st Qu.:0.0   1st Qu.:0.0   1st Qu.:3.0
 Median :3.045   Median :17.82   Median :0.5   Median :0.5   Median :3.5
 Mean   :2.988   Mean   :18.13   Mean   :0.5   Mean   :0.5   Mean   :3.5
 3rd Qu.:3.384   3rd Qu.:19.23   3rd Qu.:1.0   3rd Qu.:1.0   3rd Qu.:4.0
 Max.   :3.460   Max.   :20.22   Max.   :1.0   Max.   :1.0   Max.   :4.0
      carb
 Min.   :1.000
 1st Qu.:1.000
 Median :1.500
```

```
'data.frame':   6 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1
 $ cyl : num  6 6 4 6 8 6
 $ disp: num  160 160 108 258 360 225
 $ hp  : num  110 110 93 110 175 105
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.7
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 .
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1
 $ am  : num  1 1 1 0 0 0
 $ gear: num  4 4 4 3 3 3
 $ carb: num  4 4 1 1 2 1
>
> # Quartiles of miles per gallon (mpg)
> quartiles_mpg <- quantile(myData$mpg)
> print("Quartiles of 'mpg':")
[1] "Quartiles of 'mpg':"
> print(quartiles_mpg)
    0%    25%    50%    75%   100%
18.100 19.275 21.000 21.300 22.800
```

Part b: Subsetting and Aggregation

```
# Load the iris dataset
data(iris)
iris_head <- head(iris)
print("First six rows of 'iris' dataset:")
print(iris_head)

# Subsetting the data to include only Sepal.Width
subset_data <- subset(iris, select = Sepal.Width)
print("Subset containing only 'Sepal.Width':")
print(subset_data)

# Aggregating to find group means by Species
group_mean <- aggregate(. ~ Species, data = iris, FUN = mean)
print("Group means by Species:")
print(group_mean)

# Aggregating to find group standard deviation by Species
group_sd <- aggregate(. ~ Species, data = iris, FUN = sd)
print("Group standard deviations by Species:")
print(group_sd)
```

Output:

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

```
> print(group_mean)
     Species Sepal.Length Sepal.Width Petal.Length Petal.Width
1     setosa        5.006       3.428        1.462       0.246
2 versicolor        5.936       2.770        4.260       1.326
3  virginica        6.588       2.974        5.552       2.026
>
> # Aggregating to find group standard deviation by Species
> group_sd <- aggregate(. ~ Species, data = iris, FUN = sd)
> print("Group standard deviations by Species:")
[1] "Group standard deviations by Species:"
> print(group_sd)
     Species Sepal.Length Sepal.Width Petal.Length Petal.Width
1     setosa    0.3524897   0.3790644    0.1736640   0.1053856
2 versicolor    0.5161711   0.3137983    0.4699110   0.1977527
3  virginica    0.6358796   0.3224966    0.5518947   0.2746501
>
```

Experiment 3

Objective: Reading and writing different types of datasets

a. Reading different types of data sets (.txt. .csv) from Web and disk and writing in file in specific disk location

b. Reading Excel data in R

c. Reading XML data in R

Reading from Disk:

- **CSV File**:

  ```
  # Reading a CSV file from a specified disk location
  my_data <- read.csv("d:/data_excel.csv")
  print(my_data)
  ```

  ```
    Name Age        City
  1  John  25    New York
  2 Alice  30 Los Angeles
  3   Bob  28     Chicago
  4 Sarah  35     Houston
  ```

- **TXT File**:

  ```
  # Reading a TXT file from a specified disk location
  my_data1 <- read.delim("d:/data_txt.txt")
  print(my_data1)
  ```

  ```
    Name Age        City
  1 John  25    New York
  2 Alice  30 Los Angeles
  3  Bob  28     Chicago
  ```

- **Reading Data from the Web**

  my_data_web<-read.delim("http://example.com/data.txt")

  head(my_data_web)

  ```
    Product Price Quantity
  1   Apple  1.50      100
  2  Banana  0.75      150
  3  Orange  1.00      120
  ```

- **Writing Data to CSV File**

  my_data <- data.frame(Name = c("James", "Emma", "Lucas"), Age = c(22, 29, 24), City = c("Miami", "Chicago", "Dallas"))

  write.csv(my_data, file = "d:/my_data_output.csv")

  File Content:

  ```csv
  Name,Age,City
  James,22,Miami
  Emma,29,Chicago
  Lucas,24,Dallas
  ```

- **Reading Excel Data**

  library(readxl)

  my_excel_data <- read_xlsx("d:/data_excel.xlsx")

  print(my_excel_data)

  Output:

  ```
    Name Age Salary
  1 John  25  50000
  2 Alice  30  60000
  3 Bob  28  55000
  ```

- **Writing Data to Excel**

  library(writexl)

  write_xlsx(list(iris_sheet = iris), path = "d:/iris_data.xlsx")

- **Reading and Writing XML Data**

  library(XML)

  my_xml_data <- xmlParse("d:/data.xml")

  print(my_xml_data)

  Output:

```
<?xml version="1.0"?>
<data>
  <item>
    <name>John</name>
    <age>25</age>
  </item>
  <item>
    <name>Alice</name>
    <age>30</age>
  </item>
</data>
```

Experiment 4

Objective: Visualizations

- a. Find the data distributions using box and scatter plot
- b. Find the outliers using plot
- c. Plot the histogram, bar chart, and pie chart on sample data

Theory and Technique:

A box plot is a chart used to display data distributions by drawing boxplots for each of them. The distribution of data is based on five statistics:

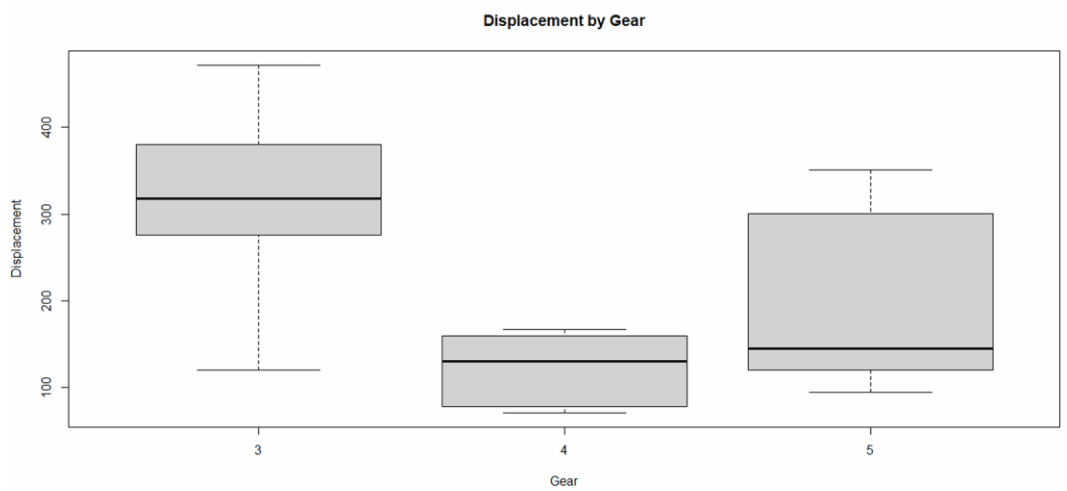
- Minimum
- First Quartile (Q1)
- Median
- Third Quartile (Q3)
- Maximum

To create a box plot, we will use the **mtcars** dataset and focus on the columns **mpg** (miles per gallon) and **cyl** (number of cylinders).

Code:

```
data(mtcars)

boxplot(disp ~ gear, data = mtcars,

    main = "Displacement by Gear",

    xlab = "Gear",

    ylab = "Displacement")
```

Output:



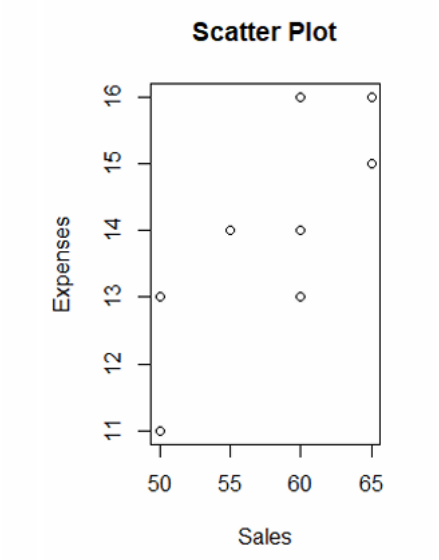
Box Plot in R

A scatter plot is a graphical representation of data points on a Cartesian plane, where each point represents a pair of values. The horizontal axis (X-axis) and vertical axis (Y-axis) represent the two variables being compared. The pattern of the points reveals if there is any correlation between the variables.

Code:

```
x =c(50,50,55,60,65,65,65,60,60,50)

 y=c(11,13,14,16,16,15,15,14,13,13)

 plot(x,y,main="Scatter Plot",xlab="Sales",ylab="Expenses")
```

Output:



Outlier Detection

Outliers are data points that significantly differ from the rest of the dataset. These points can skew statistical analyses and may indicate variability in measurement, errors, or novel patterns.

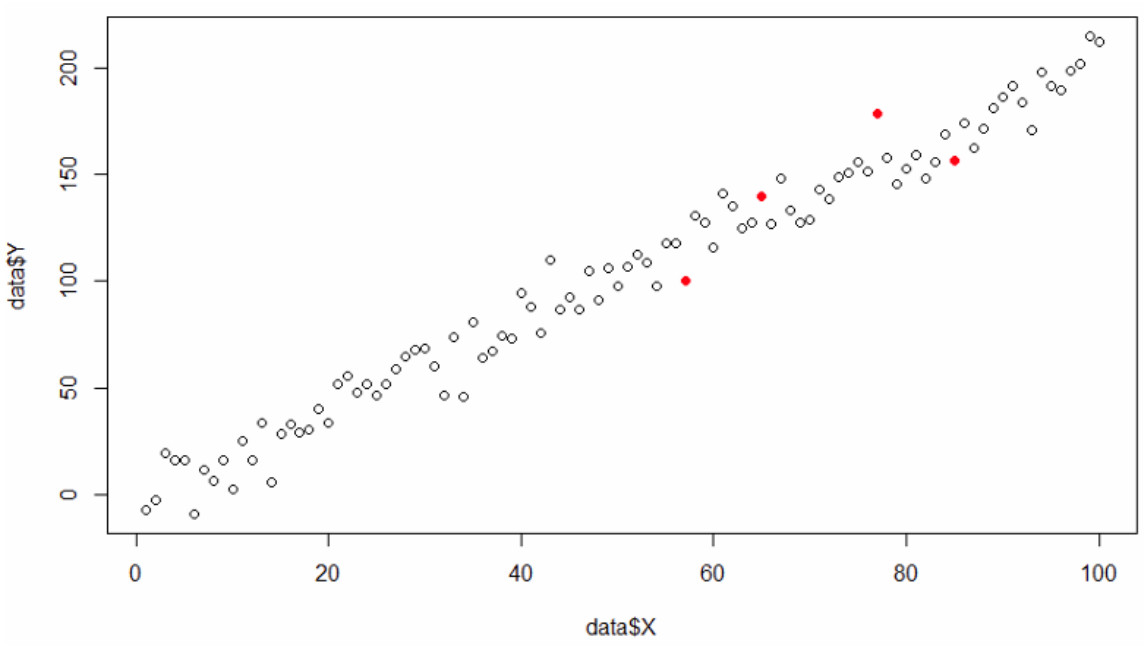
Outliers can be classified into two types based on their influence on a regression model:

1. **Leverage Points**:
    - These are data points that have an extreme value for the independent variable (X) compared to the rest of the data. Leverage points are observations that are far away from the mean of the X values but may not necessarily influence the model fit significantly.
    - Example: A data point with an X-value far from the center of the X-values in the dataset.
2. **Influential Points**:
    - These are points that have a large effect on the slope of the regression line. Influential points can disproportionately affect the results of regression analysis. A point is considered influential if removing it from the dataset changes the model significantly.
    - Example: A data point that, if removed, results in a large change in the coefficients or fit of the model.

Code:

```
data <- data.frame(x,y)

 plot(data$x, data$y)

 # Example: Detecting outliers
```

```
# Identify observations with high residuals
outliers <- which(abs(resid(mod)) > 2 * sd(resid(mod)))
X<- 1:100
Y<- 2*X+rnorm(100, mean = 0, sd = 10)
model <- lm(Y ~ X, data = data)
data <- data.frame(X = 1:100, Y = 2 * X + rnorm(100, mean = 0, sd = 10))
outliers <- which(abs(resid(model)) > 2 * sd(resid(model)))
plot(data$X, data$Y)
points(data$X[outliers], data$Y[outliers], col = "red", pch = 19)
```



Plotting Histogram, Bar Chart, and Pie Chart in R

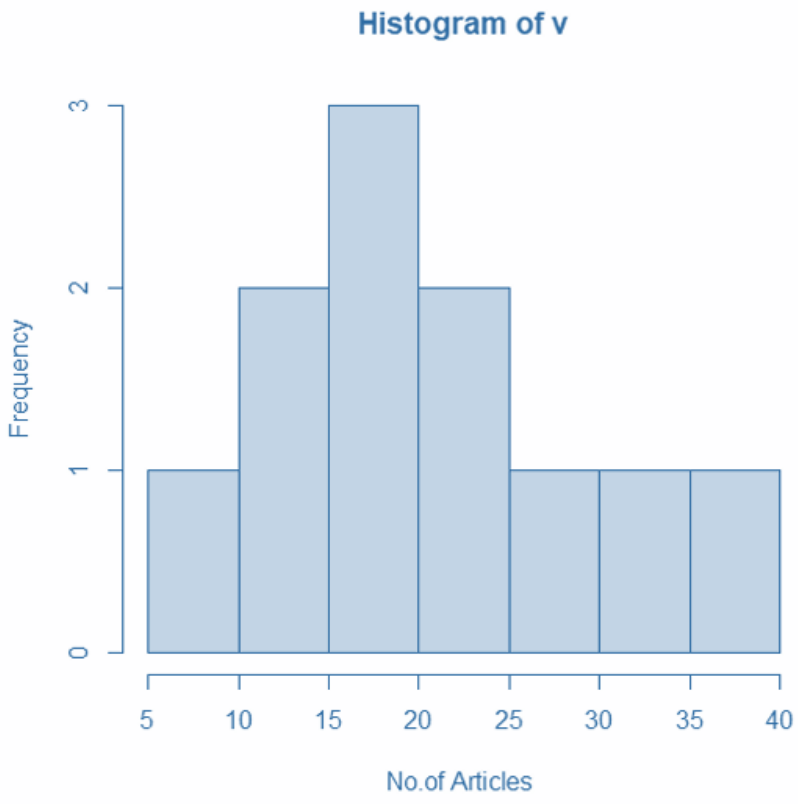
A **histogram** is used to represent the distribution of numerical data by grouping it into bins or intervals. The width of each bin corresponds to the range of data it

represents, and the height corresponds to the frequency of data points in that range.

In R, the hist() function is used to create histograms.

Code:

```
# Create data for the graph.
 v <- c(19, 23, 11, 5, 16, 21, 32, 14, 19, 27, 39)
 # Create the histogram.
 hist(v, xlab = "No.of Articles ", col = "green", border = "black")
```



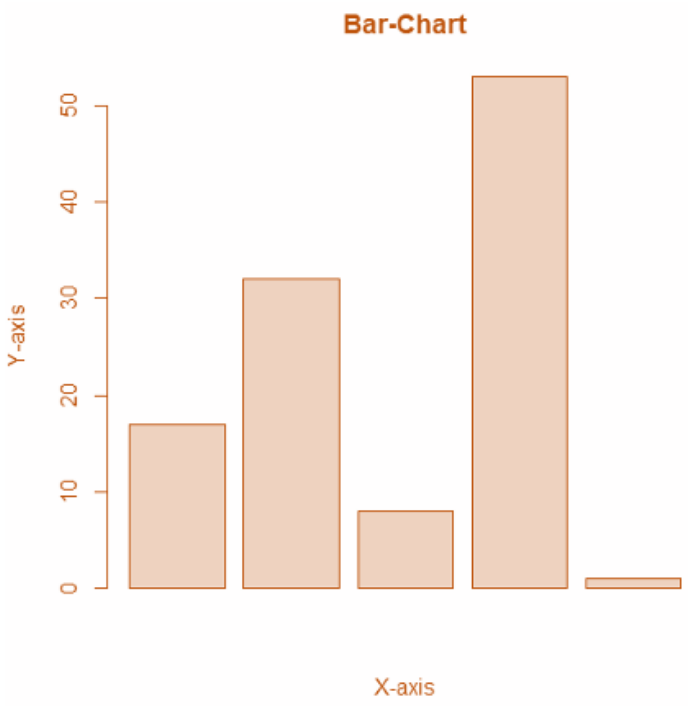
Bar Chart

A bar chart also known as bar graph is a pictorial representation of data that presents categorical data with rectangular bars with heights or lengths

proportional to the values that they represent. In other words, it is the pictorial representation of the dataset. These data sets contain the numerical values of variables that represent the length or height. R uses the barplot() function to create bar charts. Here, both vertical and Horizontal bars can be drawn.

Code:

```
# Create the data for the chart
 A<- c(17, 32, 8, 53, 1)
 # Plot the bar chart
 barplot(A, xlab = "X-axis", ylab = "Y-axis", main ="Bar-Chart")
```

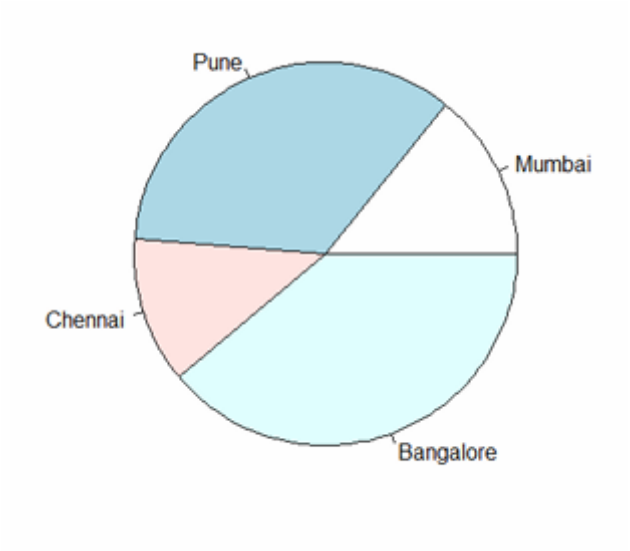


Pie Chart A pie chart is a circular statistical graphic, which is divided into slices to illustrate numerical proportions. It depicts a special chart that uses “pie slices”, where each sector shows the relative sizes of data. A circular chart cuts in the

form of radii into segments describing relative frequencies or magnitude also known as a circle graph.

Code:

```
# Create data for the graph.
 geeks<- c(23, 56, 20, 63)
 labels <- c("Mumbai", "Pune", "Chennai", "Bangalore")
 # Plot the chart.
 pie(geeks, labels)
```



Experiment 5

Objective:

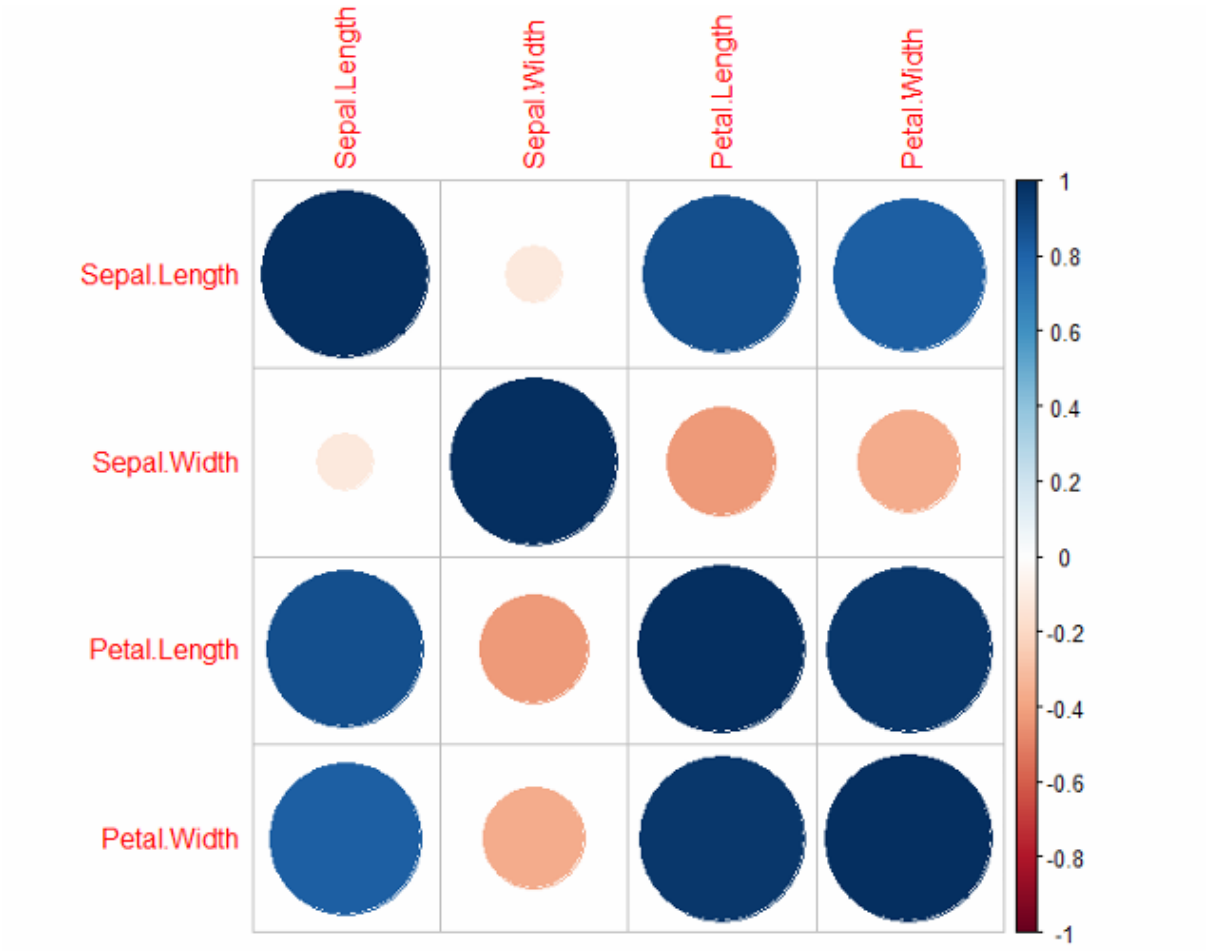
Find the correlation matrix. Plot the correlation plot on the data set and visualize giving an overview of relationships among data on iris data. Analysis of covariance: variance (ANOVA). If data have categorical variables on iris data.

Code:

```
# Load necessary packages
 install.packages("corrplot")
 library(corrplot)
 # Load the iris dataset
 data(iris)
 # Find the correlation matrix
 numeric_data <- iris[, 1:4]
 cor_matrix <- cor(numeric_data)
 print(cor_matrix)
 # Plot the correlation matrix
 corrplot(cor_matrix, method = "circle")
 # Perform ANOVA
 anova_sepal_length <- aov(Sepal.Length ~ Species, data = iris)
 summary(anova_sepal_length)
 anova_sepal_width <- aov(Sepal.Width ~ Species, data = iris)
```

```
summary(anova_sepal_width)

anova_petal_length <- aov(Petal.Length ~ Species, data = iris)

summary(anova_petal_length)

anova_petal_width <- aov(Petal.Width ~ Species, data = iris)

summary(anova_petal_width)
```

Output:



Experiment 6

Objective:

Import data from web storage. Name the dataset and now do logistic regression to find out the relation between variables that are affecting the admission of a student in an institute based on his or her GRE score, GPA obtained, and rank of the student. Also, the model is fit or not. Require (foreign), require (MASS).

Code:

```
require(foreign)

require(MASS)

# Import the dataset (Assuming it's a CSV file)

data <- read.csv("D:/gre_admission_data.csv")

# Check the first few rows of the dataset

head(data)

# Convert Admitted to factor type

data$Admitted <- as.factor(data$Admitted)

# Check the structure of the data

str(data)

# Fit the logistic regression model

log_model <- glm(Admitted ~ GRE + GPA + Rank, data = data, family = binomial)


# Summary of the logistic regression model
```

summary(log_model)

Output:

```
> head(admission_data)
  GRE.Score TOEFL.Score University.Rating SOP LOR CGPA Research Chance.of.Admit
1       337         118                 4 4.5 4.5 9.65        1             0.92
2       324         107                 4 4.0 4.5 8.87        1             0.76
3       316         104                 3 3.0 3.5 8.00        1             0.72
4       322         110                 3 3.5 2.5 8.67        1             0.80
5       314         103                 2 2.0 3.0 8.21        0             0.65
6       330         115                 5 4.5 3.0 9.34        1             0.90
>
> # Check the structure of the dataset
> str(admission_data)
'data.frame':   500 obs. of  8 variables:
 $ GRE.Score        : int  337 324 316 322 314 330 321 308 302 323 ...
 $ TOEFL.Score      : int  118 107 104 110 103 115 109 101 102 108 ...
 $ University.Rating: int  4 4 3 3 2 5 3 2 1 3 ...
 $ SOP              : num  4.5 4 3 3.5 2 4.5 3 3 2 3.5 ...
 $ LOR              : num  4.5 4.5 3.5 2.5 3 3 4 4 1.5 3 ...
 $ CGPA             : num  9.65 8.87 8 8.67 8.21 9.34 8.2 7.9 8 8.6 ...
 $ Research         : int  1 1 1 1 0 1 1 0 0 0 ...
 $ Chance.of.Admit  : num  0.92 0.76 0.72 0.8 0.65 0.9 0.75 0.68 0.5 0.45 ...
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -50.87887    9.71773  -5.236 1.64e-07 ***
GRE.Score     0.06802    0.03202   2.125   0.0336 *
CGPA          4.13468    0.78050   5.297 1.17e-07 ***
rank2        -1.27097    0.62590  -2.031   0.0423 *
rank3        -0.87239    0.73000  -1.195   0.2321
rank4        -0.56899    1.04590  -0.544   0.5864
rank5        12.51404  973.72338   0.013   0.9897
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 263.86  on 499  degrees of freedom
Residual deviance: 146.05  on 493  degrees of freedom
AIC: 160.05

Number of Fisher Scoring iterations: 18
```

```
> summary(logistic_model)$deviance
[1] 146.0529
> summary(logistic_model)$null.deviance
[1] 263.8649
> summary(logistic_model)$aic
[1] 160.0529
>
```

```
> print(confusion_matrix)
         Actual
Predicted   0   1
        0  13   5
        1  24 458
>
> # Calculate accuracy
> accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
> print(paste("Accuracy:", accuracy))
[1] "Accuracy: 0.942"
```

Experiment 7

Objective: Apply multiple regressions, if data have a continuous dependent variable. Apply to the above dataset.

Code:

```
# Replace with your local file path
 file_path <- "C:\\Users\\yashm\\Desktop\\MLDA Lab\\admission_data.csv"
 # Import the dataset
 admission_data <- read.csv(file_path)
 # Convert 'rank' to a factor
 admission_data$rank <- as.factor(admission_data$University.Rating)
 # Set the threshold
 threshold <- 0.5
 # Convert probabilities to binary outcomes
 admission_data$admit <- ifelse(admission_data$Chance.of.Admit >= threshold, 1, 0)
 # Fit the multiple regression model
 regression_model <- lm(admit ~ GRE.Score + CGPA + rank, data = admission_data)
 summary(regression_model)
```

Output:

```
Residuals:
     Min       1Q   Median       3Q      Max
-0.98811 -0.03344  0.03825  0.13038  0.34114

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.166777   0.384904  -3.031  0.00256 **
GRE.Score    0.001619   0.001688   0.959  0.33799
CGPA         0.184361   0.034231   5.386 1.12e-07 ***
rank2       -0.000193   0.046743  -0.004  0.99671
rank3        0.046352   0.047922   0.967  0.33390
rank4       -0.023365   0.054714  -0.427  0.66954
rank5       -0.074686   0.061424  -1.216  0.22460
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2367 on 493 degrees of freedom
Multiple R-squared:  0.1935,    Adjusted R-squared:  0.1837
F-statistic: 19.71 on 6 and 493 DF,  p-value: < 2.2e-16
```

Experiment 8

Objective: Regression Model for Prediction Apply regression Model techniques to predict the data on above dataset.

Code:

```
#install.packages("ggplot2")
 #install.packages("dplyr")
 #install.packages("caret")
 #install.packages("pROC")
 library(ggplot2)
 library(dplyr)
 library(caret)
 library(pROC)
 # Replace with your local file path
 file_path <- "C:\\Users\\yashm\\Desktop\\MLDA Lab\\admission_data.csv"
 # Import the dataset
 admission_data <- read.csv(file_path)
 # Convert 'rank' to a factor
 admission_data$rank <- as.factor(admission_data$University.Rating)
 # Set the threshold
 threshold <- 0.5
 # Convert probabilities to binary outcomes
```

```
admission_data$admit <- ifelse(admission_data$Chance.of.Admit >= threshold, 1, 0)

# Fit the multiple regression model

regression_model <- lm(admit ~ GRE.Score + CGPA + rank, data = admission_data)

summary(regression_model)

# Make predictions on the dataset

admission_data$predicted_probabilities <- predict(regression_model)

# Convert probabilities to binary outcomes using a threshold (e.g., 0.5)

admission_data$predicted_admit<ifelse(admission_data$predicted_probabilities >= 0.5, 1, 0)

admission_data$predicted_admit <- as.factor(admission_data$predicted_admit)

admission_data$admit <- as.factor(admission_data$admit)

# View the first few rows of the dataset with predictions

head(admission_data)

# Create confusion matrix

confusion_matrix <- confusionMatrix(as.factor(admission_data$predicted_admit),

as.factor(admission_data$admit))

print(confusion_matrix)

# Compute ROC curve

roc_curve<-roc(admission_data$admit, dmission_data$predicted_probabilities)

plot(roc_curve)
```

```
# Compute AUC

auc_value <- auc(roc_curve)

print(paste("AUC:", auc_value))
```

Output:

```
Residuals:
     Min       1Q   Median       3Q      Max
-0.98811 -0.03344  0.03825  0.13038  0.34114

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.166777   0.384904  -3.031  0.00256 **
GRE.Score    0.001619   0.001688   0.959  0.33799
CGPA         0.184361   0.034231   5.386 1.12e-07 ***
rank2       -0.000193   0.046743  -0.004  0.99671
rank3        0.046352   0.047922   0.967  0.33390
rank4       -0.023365   0.054714  -0.427  0.66954
rank5       -0.074686   0.061424  -1.216  0.22460
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2367 on 493 degrees of freedom
Multiple R-squared:  0.1935,    Adjusted R-squared:  0.1837
F-statistic: 19.71 on 6 and 493 DF,  p-value: < 2.2e-16
```

```
> print(confusion_matrix)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0   0   0
         1  37 463

               Accuracy : 0.926
                 95% CI : (0.8994, 0.9474)
    No Information Rate : 0.926
    P-Value [Acc > NIR] : 0.5436

                  Kappa : 0

 Mcnemar's Test P-Value : 3.252e-09
```

Experiment 9

Objective: Classification Model

a. Install relevant package for classification.

b. Choose classifier for classification problem.

c. Evaluate the performance of classifier.

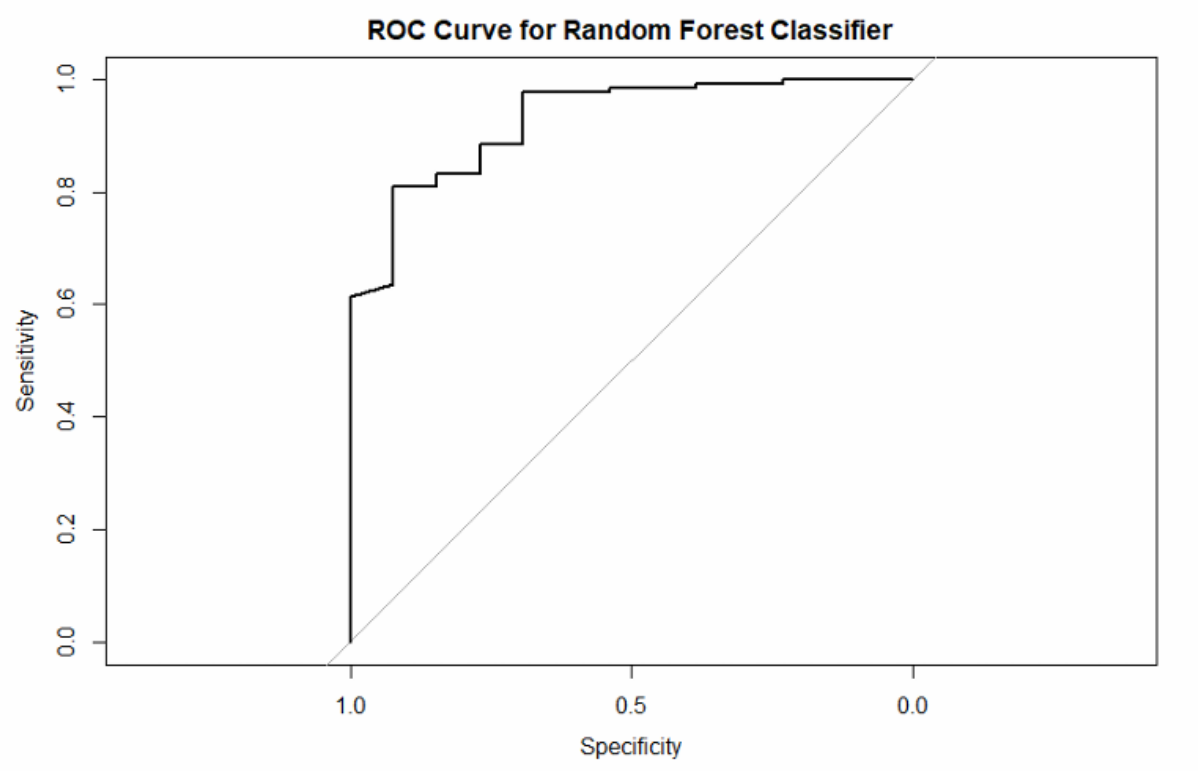
Code:

```
# Install and load relevant packages
 library(caret)
 library(randomForest)
 library(pROC)
 file_path <- "C:\\Users\\yashm\\Desktop\\MLDA Lab\\admission_data.csv"
admission_data <- read.csv(file_path)
 admission_data$rank <- as.factor(admission_data$University.Rating)
 threshold <- 0.5
admission_data$admit <- ifelse(admission_data$Chance.of.Admit >= threshold, 1, 0)
set.seed(123)
 train_index <- createDataPartition(admission_data$admit, p = 0.7, list = FALSE)
 train_data <- admission_data[train_index, ]
 test_data <- admission_data[-train_index, ]
 rf_model <- randomForest(admit ~ GRE.Score + CGPA + rank, data = train_data, importance = TRUE)
```

```
rf_predictions <- predict(rf_model, newdata = test_data)

rf_bin_predictions <- ifelse(rf_predictions >= threshold, 1, 0)

 conf_matrix <- confusionMatrix(as.factor(rf_bin_predictions),

 as.factor(test_data$admit))

 print(conf_matrix)

 roc_curve <- roc(test_data$admit, as.numeric(rf_predictions))

 plot(roc_curve, main = "ROC Curve for Random Forest Classifier")
```

Output:

```
> print(conf_matrix)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0   7   3
         1   6 134

               Accuracy : 0.94
                 95% CI : (0.8892, 0.9722)
    No Information Rate : 0.9133
    P-Value [Acc > NIR] : 0.154

                  Kappa : 0.5768

 Mcnemar's Test P-Value : 0.505

            Sensitivity : 0.53846
            Specificity : 0.97810
         Pos Pred Value : 0.70000
         Neg Pred Value : 0.95714
             Prevalence : 0.08667
         Detection Rate : 0.04667
   Detection Prevalence : 0.06667
      Balanced Accuracy : 0.75828

       'Positive' Class : 0
```


ROC Curve for Random Forest Classifier
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1.0
0.5
0.0
Specificity

Experiment 10

Objective: Clustering Model

a. Clustering algorithms for unsupervised classification.

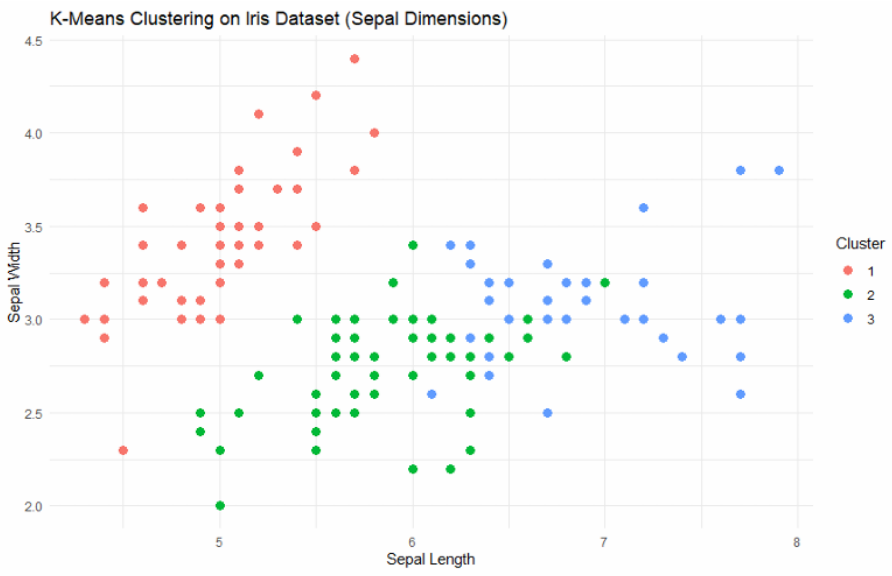
b. Plot the cluster data using R visualizations.

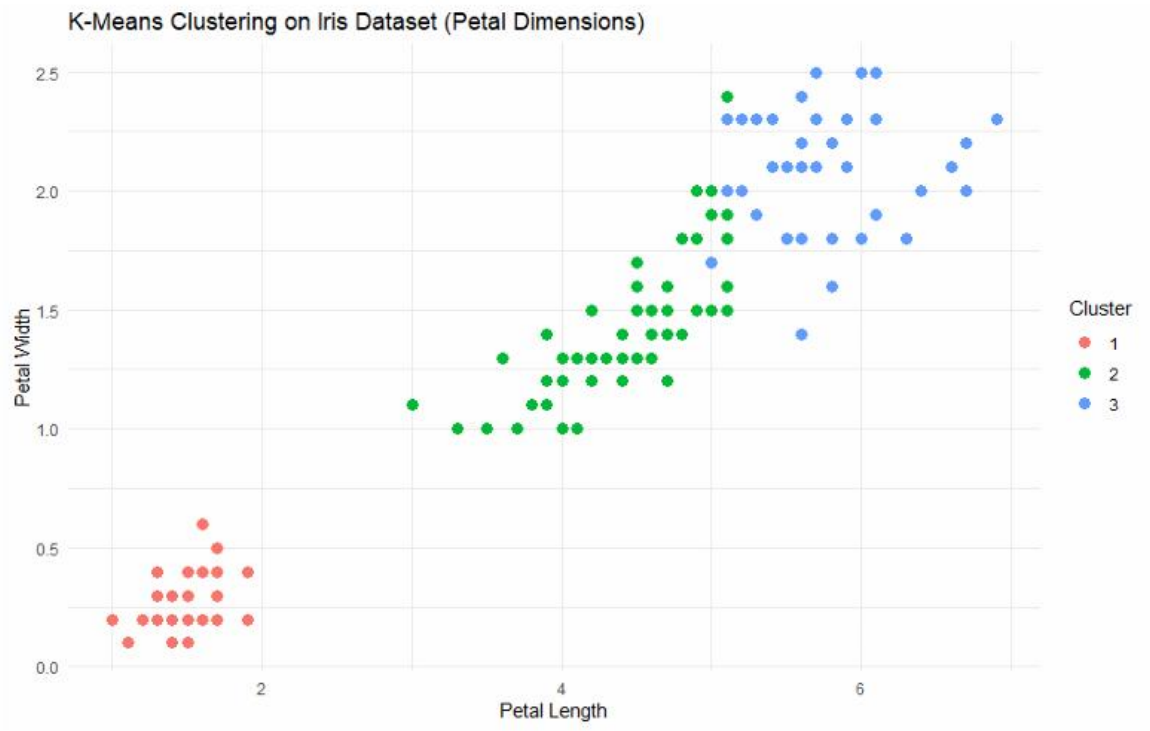
Code:

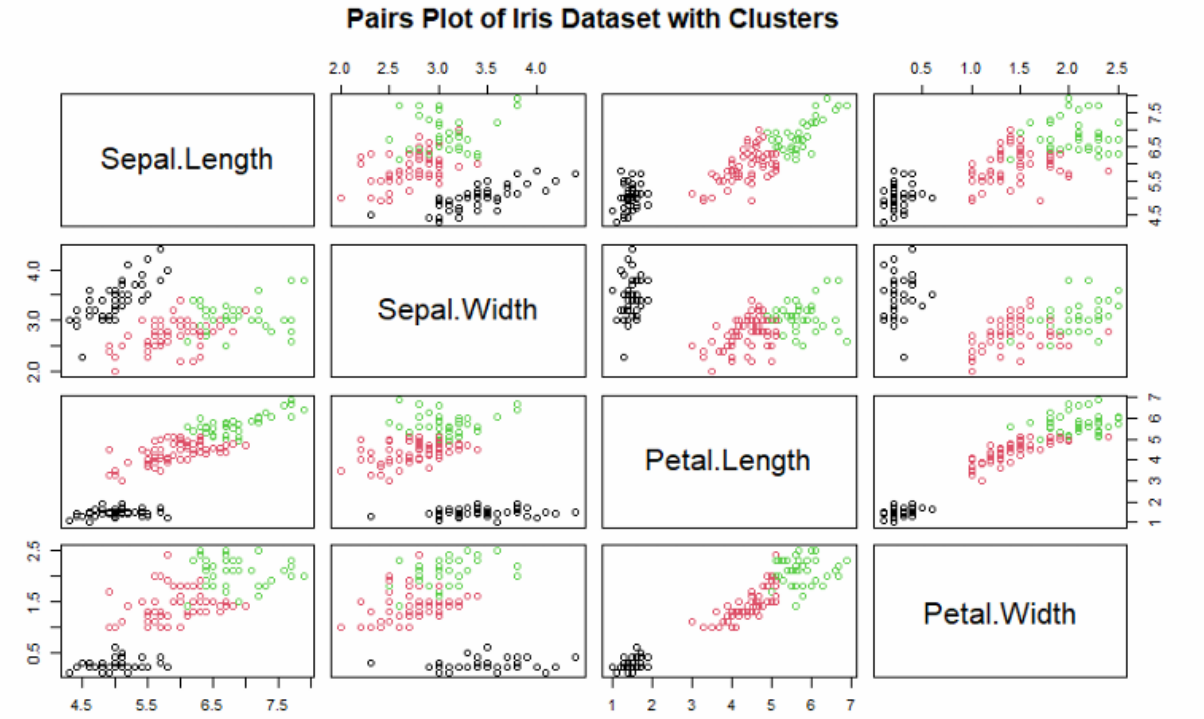
```
# Load necessary libraries
library(ggplot2)
# Load the iris dataset
data(iris)
# Set seed for reproducibility
set.seed(123)
# Perform k-means clustering with 3 clusters
kmeans_result <- kmeans(iris[, 1:4], centers = 3)
# Add the cluster assignment to the iris dataset
iris$Cluster <- as.factor(kmeans_result$cluster)
# Visualize Sepal dimensions (Sepal.Length vs Sepal.Width) by Cluster
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Cluster)) +
geom_point(size = 3) +
labs(title = "K-Means Clustering on Iris Dataset (Sepal Dimensions)",
x = "Sepal Length", y = "Sepal Width") +
theme_minimal()
# Visualize Petal dimensions (Petal.Length vs Petal.Width) by Cluster
```

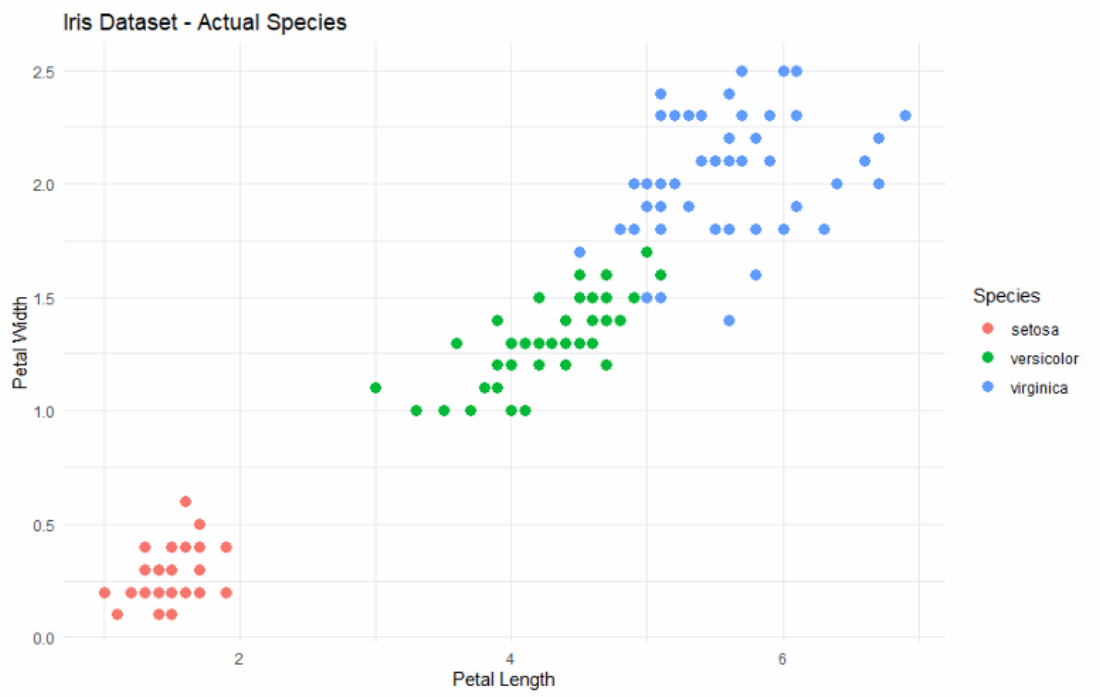
```
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Cluster)) +
geom_point(size = 3) +
labs(title = "K-Means Clustering on Iris Dataset (Petal Dimensions)",
x = "Petal Length", y = "Petal Width") +
theme_minimal()
# Pairs plot with clusters
pairs(iris[, 1:4], col = iris$Cluster, main = "Pairs Plot of Iris Dataset with Clusters")
# Compare with actual species labels
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Species)) +
geom_point(size = 3) +
labs(title = "Iris Dataset- Actual Species",
x = "Petal Length", y = "Petal Width") +
theme_minimal()
```

Output:




K-Means Clustering on Iris Dataset (Petal Dimensions)
Petal Width
Petal Length
Cluster
1
2
3
0.0
0.5
1.0
1.5
2.0
2.5
2
4
6


Pairs Plot of Iris Dataset with Clusters
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width


Iris Dataset - Actual Species
Petal Width
Petal Length
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
Species
setosa
versicolor
virginica