Lab Observation Book

R V Abhishek

2025-08-13

B.M.S. COLLEGE OF ENGINEERING

(Autonomous College under VTU)

Bull Temple Road, Basavangudi, Bangalore - 560019

Lab Observation

ON

Programming with R

Submitted by

R V Abhishek(1BM23CD047)

in fulfillment of mandatory observation submission for Lab assessment

BACHELOR OF ENGINEERING

in

Computer Science & Engineering (Data Science)

Under the Guidance of

Dr. Kalyan N

Assistant Professor

Department of CSE (Data Science),

B.M.S. College of Engineering

2025-2026

B.M.S. COLLEGE OF ENGINEERING

(Autonomous College under VTU)

Bull Temple Road, Basavangudi, Bangalore - 560019

Laboratory Certificate

This is to certify that Mr./Ms. **R V Abhishek** has satisfactorily completed the course of experiments in practical **Programming With R** prescribed by the Visvesvaraya Technology University for 5^{th} Semester Bachelor of Engineering course in the laboratory of the college in the year 2024 - 2025

Head of the Department

Staff Incharge of the Batch

Marks	
Maximum	**Obtained**

Name of the Candidate: R V Abhishek

Branch: CSE (Data Science)

USN: 1BM23CD047

Date:

Signature of the Candidate

TABLE OF CONTENTS

Program - 1

Program to check what type of Triangle given 3 sides, and calculate its area

Date of Execution - 2025-08-13

This R program validates the sides of the triangle (taken as input from the user) and then if valid, calculates the area of the triangle using Heron's formula and checks what type of triangle it is

```r
# Validating the triangle
is_valid_triangle <- function(a, b, c) {
  return ((a + b > c) & (b + c > a) & (a + c > b ))
}
```

```r
# Function to check the type of triangle
triangle_type <- function(a , b , c) {
  if (a == b && b == c) {
    return(" Equilateral ")
  } else if ( a == b || b == c || a == c) {
    return(" Isosceles ")
  } else {
    return("Scalene")
  }
}
```

```r
# Calculating Area using Heron's Formula
triangle_area <- function(a , b , c) {
  s <- (a + b + c) / 2 # Semi - perimeter
  # Heron 's formula
  area <- sqrt (s * (s - a) * (s - b) * (s - c))
  return (area)
}
```

```r
# Validating inputs
validate_input <- function(x) {
  if (!is.numeric(x) || x <= 0) {
    stop("Error : Input must be a positive number.")
  }
  return(TRUE)
}
```

```r
## Main Code Block

# 1. Defining 3 variables representing the 3 sides of the triangle
cat("Enter the lengths of the sides of the triangle :\n")
```

```
Enter the lengths of the sides of the triangle :
```

```r
a <- as.numeric(readline(prompt = "Side A: "))
```

```
Side A:
```

```r
b <- as.numeric(readline(prompt = "Side B: "))
```

```
Side B:
```

```r
c <- as.numeric(readline(prompt = "Side C: "))
```

```
Side C:
```

```r
# 2.  Input Validation and implementation of all the functions.
# Input validation}
tryCatch ({
  validate_input(a)
  validate_input(b)
  validate_input(c)

  # Check if the inputs form a valid triangle
  if (!is_valid_triangle(a , b , c)) {
    stop("Error : The given sides do not form a valid triangle.")
  }

  # Determine the type of triangle
  type_of_triangle <- triangle_type(a , b , c)
  cat("The triangle is:", type_of_triangle, "\n")
  # Calculate the area of the triangle
  area_of_triangle <- triangle_area(a, b, c)
  cat("The area of the triangle is:", area_of_triangle, "\n")

}, error = function(e){
  cat(e$message, "\n")
})
```

```
missing value where TRUE/FALSE needed
```

Sample Output

```
Enter the lengths of the sides of the triangle:
Side a: 5
Side b: 5
Side c: 8

The triangle is: Isosceles
The area of the triangle is: 12

Enter the lengths of the sides of the triangle:**
Side a: 1
Side b: 2
Side c: 8

Error: The given sides do not form a valid triangle.
```

Program - 2

Creating and Manipulating Data Structures

Date of Execution - 2025-08-20

Objective - This program evaluates the student's understanding of different data structures (vectors, matrices, lists, and data frames) in R and how to manipulate them.

```r
# 1. Create a vector of random numbers and apply operations such as sorting and searching


set.seed(42) # For reproducibility
random_vector <- runif(20, min = 1, max = 100)
cat("Original random vector:\n")
```

```
Original random vector:
```

```r
print(random_vector)
```

```
 [1] 91.56580 93.77047 29.32781 83.21431 64.53281 52.39050 73.92224
 [8] 14.33199 66.04224 70.80141 46.31644 72.19211 93.53255 26.28745
[15] 46.76699 94.06144 97.84442 12.63125 48.02471 56.47294
```

```r
# Sort the vector
sorted_vector <- sort(random_vector)
cat("Sorted vector:\n")
```

```
Sorted vector:
```

```r
print(sorted_vector)
```

```
 [1] 12.63125 14.33199 26.28745 29.32781 46.31644 46.76699 48.02471
 [8] 52.39050 56.47294 64.53281 66.04224 70.80141 72.19211 73.92224
[15] 83.21431 91.56580 93.53255 93.77047 94.06144 97.84442
```

```r
# Search for a specific value (check if a number is present)
search_value <- 50
is_value_present <- any(random_vector == search_value)
cat("Is", search_value, "present in the vector?", is_value_present, "\n")
```

```
Is 50 present in the vector? FALSE
```

```r
# Find values in the vector greater than 60
values_greater_than_60 <- random_vector[random_vector > 60]
cat("Values greater than 60:\n")
```

```
Values greater than 60:
```

```r
print(values_greater_than_60)
```

```
 [1] 91.56580 93.77047 83.21431 64.53281 73.92224 66.04224 70.80141
 [8] 72.19211 93.53255 94.06144 97.84442
```

```r
# 2. Convert the vector into a matrix and perform matrix multiplication

# Create a 4x5 matrix from the vector
matrix_from_vector <- matrix(random_vector, nrow = 4, ncol = 5)
cat("Matrix from vector:\n")
```

```
Matrix from vector:
```

```r
print(matrix_from_vector)
```

```
         [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 91.56580 64.53281 66.04224 93.53255 97.84442
[2,] 93.77047 52.39050 70.80141 26.28745 12.63125
[3,] 29.32781 73.92224 46.31644 46.76699 48.02471
[4,] 83.21431 14.33199 72.19211 94.06144 56.47294
```

```r
# Perform matrix multiplication (matrix with its transpose)
matrix_transpose <- t(matrix_from_vector)
matrix_multiplication_result <- matrix_from_vector %*% matrix_transpose
cat("Matrix multiplication result:\n")
```

```
Matrix multiplication result:
```

```r
print(matrix_multiplication_result)
```

```
         [,1]     [,2]     [,3]     [,4]
[1,] 35232.22 20337.59 19587.86 27635.57
[2,] 20337.59 17401.08 11738.17 16851.17
[3,] 19587.86 11738.17 12963.36 13954.70
[4,] 27635.57 16851.17 13954.70 24378.48
```

```r
# Element-wise matrix multiplication (Hadamard product)
elementwise_multiplication_result <- matrix_from_vector * matrix_from_vector
cat("Element-wise matrix multiplication result:\n")
```

```
Element-wise matrix multiplication result:
```

```r
print(elementwise_multiplication_result)
```

```
          [,1]     [,2]     [,3]      [,4]      [,5]
[1,] 8384.2954 4164.483 4361.577 8748.3384 9573.5298
[2,] 8792.9003 2744.764 5012.840  691.0302  159.5484
[3,]  860.1207 5464.498 2145.212 2187.1513 2306.3729
[4,] 6924.6222  205.406 5211.701 8847.5541 3189.1932
```

```r
# 3. Create a list containing different types of elements and perform subsetting

my_list <- list(
  numbers = random_vector,
  characters = c("A", "B", "C", "D"),
  logical_values = c(TRUE, FALSE, TRUE),
  matrix = matrix_from_vector
)
cat("List:\n")
```

```
List:
```

```r
print(my_list)
```

```
$numbers
 [1] 91.56580 93.77047 29.32781 83.21431 64.53281 52.39050 73.92224
 [8] 14.33199 66.04224 70.80141 46.31644 72.19211 93.53255 26.28745
[15] 46.76699 94.06144 97.84442 12.63125 48.02471 56.47294

$characters
[1] "A" "B" "C" "D"

$logical_values
[1]  TRUE FALSE  TRUE

$matrix
         [,1]     [,2]     [,3]     [,4]     [,5]
```

```
[1,] 91.56580 64.53281 66.04224 93.53255 97.84442
[2,] 93.77047 52.39050 70.80141 26.28745 12.63125
[3,] 29.32781 73.92224 46.31644 46.76699 48.02471
[4,] 83.21431 14.33199 72.19211 94.06144 56.47294
```

```r
# Subsetting the list (extracting numeric and logical parts)
subset_numeric <- my_list$numbers
cat("Subset (numeric part of the list):\n")
```

```
Subset (numeric part of the list):
```

```r
str(subset_numeric)
```

```
 num [1:20] 91.6 93.8 29.3 83.2 64.5 ...
```

```r
cat("\n")
```

```r
subset_logical <- my_list$logical_values
cat("Subset (logical part of the list):\n", subset_logical, "\n")
```

```
Subset (logical part of the list):
 TRUE FALSE TRUE
```

```r
# Modify elements in the list (replace the second character with "Z")
my_list$characters[2] <- "Z"
cat("Modified list of characters:\n", my_list$characters, "\n")
```

```
Modified list of characters:
 A Z C D
```

```r
# Apply a function to the numeric part of the list
# (e.g., calculate the square of the numbers)
squared_numbers <- my_list$numbers ^ 2
cat("Squared numbers:\n")
```

```
Squared numbers:
```

```r
str(squared_numbers)
```

```
 num [1:20] 8384 8793 860 6925 4164 ...
```

```r
cat("\n")
```

```r
# 4. Create a data frame and perform operations such as filtering,
# summarizing, and handling missing values
# Create a data frame
df <- data.frame(
  ID = 1:20,
  Age = sample(18:65, 20, replace = TRUE),
  Score = runif(20, min = 50, max = 100),
  Passed = sample(c(TRUE, FALSE), 20, replace = TRUE)
)
cat("Data frame:\n")
```

```
Data frame:
```

```r
print(df)
```

```
   ID Age    Score Passed
1   1  64 71.78858  FALSE
2   2  20 51.87155  FALSE
3   3  58 98.67700  FALSE
4   4  42 71.58756  FALSE
5   5  44 97.87883  FALSE
6   6  53 94.38775  FALSE
7   7  54 81.99894   TRUE
8   8  48 98.54833  FALSE
9   9  62 80.94191   TRUE
10 10  22 66.67136  FALSE
11 11  37 67.33741   TRUE
12 12  51 69.92427  FALSE
13 13  45 89.23464  FALSE
14 14  57 51.94682  FALSE
15 15  20 87.43977  FALSE
16 16  50 83.86384   TRUE
17 17  59 58.56322   TRUE
18 18  41 63.05440   TRUE
19 19  47 75.72065   TRUE
20 20  60 83.78036  FALSE
```

```r
# Filter the data frame (rows where Age > 30 and Score > 70)
filtered_df <- subset(df, Age > 30 & Score > 70)
cat("Filtered data frame (Age > 30 and Score > 70):\n")
```

```
Filtered data frame (Age > 30 and Score > 70):
```

```r
print(filtered_df)
```

```
   ID Age    Score Passed
1   1  64 71.78858  FALSE
3   3  58 98.67700  FALSE
4   4  42 71.58756  FALSE
5   5  44 97.87883  FALSE
6   6  53 94.38775  FALSE
7   7  54 81.99894   TRUE
8   8  48 98.54833  FALSE
9   9  62 80.94191   TRUE
13 13  45 89.23464  FALSE
16 16  50 83.86384   TRUE
19 19  47 75.72065   TRUE
20 20  60 83.78036  FALSE
```

```r
# Calculate mean, sum, and variance of numerical columns (Age and Score)
mean_age <- mean(df$Age)
sum_age <- sum(df$Age)
var_age <- var(df$Age)

mean_score <- mean(df$Score)
sum_score <- sum(df$Score)
var_score <- var(df$Score)

cat("Summary statistics for Age column:\n")
```

```
Summary statistics for Age column:
```

```r
cat("Mean Age:", mean_age, "\n")
```

```
Mean Age: 46.7
```

```r
cat("Sum of Age:", sum_age, "\n")
```

```
Sum of Age: 934
```

```r
cat("Variance of Age:", var_age, "\n")
```

```
Variance of Age: 179.6947
```

```r
cat("Summary statistics for Score column:\n")
```

```
Summary statistics for Score column:
```

```r
cat("Mean Score:", mean_score, "\n")
```

```
Mean Score: 77.26086
```

```r
cat("Sum of Score:", sum_score, "\n")
```

```
Sum of Score: 1545.217
```

```r
cat("Variance of Score:", var_score, "\n")
```

```
Variance of Score: 219.2162
```

```r
# 5. Handling missing values in the data frame

# Introduce some NA values in the Score column
df$Score[sample(1:20, 5)] <- NA
cat("Data frame with missing values:\n")
```

```
Data frame with missing values:
```

```r
print(df)
```

```
   ID Age    Score Passed
1   1  64 71.78858  FALSE
2   2  20 51.87155  FALSE
3   3  58 98.67700  FALSE
4   4  42       NA  FALSE
5   5  44 97.87883  FALSE
6   6  53 94.38775  FALSE
```

```
7   7  54 81.99894   TRUE
8   8  48 98.54833  FALSE
9   9  62       NA   TRUE
10 10  22       NA  FALSE
11 11  37 67.33741   TRUE
12 12  51       NA  FALSE
13 13  45       NA  FALSE
14 14  57 51.94682  FALSE
15 15  20 87.43977  FALSE
16 16  50 83.86384   TRUE
17 17  59 58.56322   TRUE
18 18  41 63.05440   TRUE
19 19  47 75.72065   TRUE
20 20  60 83.78036  FALSE
```

```r
# Replace NA values with the mean of the Score column
df$Score[is.na(df$Score)] <- mean(df$Score, na.rm = TRUE)
cat("Data frame after imputation of missing values:\n")
```

```
Data frame after imputation of missing values:
```

```r
print(df)
```

```
   ID Age    Score Passed
1   1  64 71.78858  FALSE
2   2  20 51.87155  FALSE
3   3  58 98.67700  FALSE
4   4  42 77.79050  FALSE
5   5  44 97.87883  FALSE
6   6  53 94.38775  FALSE
7   7  54 81.99894   TRUE
8   8  48 98.54833  FALSE
9   9  62 77.79050   TRUE
10 10  22 77.79050  FALSE
11 11  37 67.33741   TRUE
12 12  51 77.79050  FALSE
13 13  45 77.79050  FALSE
14 14  57 51.94682  FALSE
15 15  20 87.43977  FALSE
16 16  50 83.86384   TRUE
17 17  59 58.56322   TRUE
```

```
18 18  41 63.05440   TRUE
19 19  47 75.72065   TRUE
20 20  60 83.78036  FALSE
```

```r
# Grouping the data by Passed status and calculating group-wise statistics
library(dplyr)
```

```

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
grouped_stats <- df %>%
  group_by(Passed) %>%
  summarise(
    mean_score = mean(Score, na.rm = TRUE),
    mean_age = mean(Age)
  )
cat("Grouped statistics by Passed status:\n")
```

```
Grouped statistics by Passed status:
```

```r
print(grouped_stats)
```

```
# A tibble: 2 x 3
  Passed mean_score mean_age
  <lgl>       <dbl>    <dbl>
1 FALSE        80.6     44.9
2 TRUE         72.6     50
```

Program - 3

Basic Statistical Operations on Open-Source Datasets

Date of Execution - 2025-08-26

Objective: This program emphasizes the application of statistical concepts on real-world datasets and visualization of the data.

```
# Load necessary
library(dplyr)  # For data manipulation
library(ggplot2)  # For visualization
library(moments)  # For skewness and kurtosis
library(palmerpenguins) # For Palmer Penguins dataset

data(iris)  # Load Iris dataset

data(penguins)  # Load Palmer Penguins
```

```
# Function to calculate mode
calc_mode <- function(x) {
  return (as.numeric (names (sort (table (x), decreasing = TRUE)) [1] ))
}
```

```
# Perform Statistical Analysis on Iris Dataset
print("----- Iris Dataset Analysis -----")
```

```
[1] "----- Iris Dataset Analysis -----"
```

```
# Mean
iris_mean <- sapply (iris[, 1:4], mean, na.rm = TRUE )
print(paste("Mean of Iris dataset : ", iris_mean))
```

```
[1] "Mean of Iris dataset :  5.84333333333333"
[2] "Mean of Iris dataset :  3.05733333333333"
[3] "Mean of Iris dataset :  3.758"
[4] "Mean of Iris dataset :  1.19933333333333"
```

```
#Median
iris_median <- sapply(iris[, 1:4], median, na.rm = TRUE )
print(paste("Median of Iris dataset : ", iris_median))
```

```
[1] "Median of Iris dataset :  5.8"  "Median of Iris dataset :  3"
[3] "Median of Iris dataset :  4.35" "Median of Iris dataset :  1.3"
```

```r
#Mode
iris_mode <- sapply(iris[, 1:4], calc_mode )
print(paste("Mode of Iris dataset : ", iris_median))
```

```
[1] "Mode of Iris dataset :  5.8"  "Mode of Iris dataset :  3"
[3] "Mode of Iris dataset :  4.35" "Mode of Iris dataset :  1.3"
```

```r
#Variance
iris_variance <- sapply(iris[, 1:4], var, na.rm = TRUE )
print(paste("Variance of Iris dataset : ", iris_variance))
```

```
[1] "Variance of Iris dataset :  0.685693512304251"
[2] "Variance of Iris dataset :  0.189979418344519"
[3] "Variance of Iris dataset :  3.11627785234899"
[4] "Variance of Iris dataset :  0.581006263982103"
```

```r
#Standard Deviation
iris_sd <- sapply(iris[, 1:4], sd, na.rm = TRUE )
print(paste("Standard Deviation of Iris dataset : ", iris_sd))
```

```
[1] "Standard Deviation of Iris dataset :  0.828066127977863"
[2] "Standard Deviation of Iris dataset :  0.435866284936698"
[3] "Standard Deviation of Iris dataset :  1.76529823325947"
[4] "Standard Deviation of Iris dataset :  0.762237668960347"
```

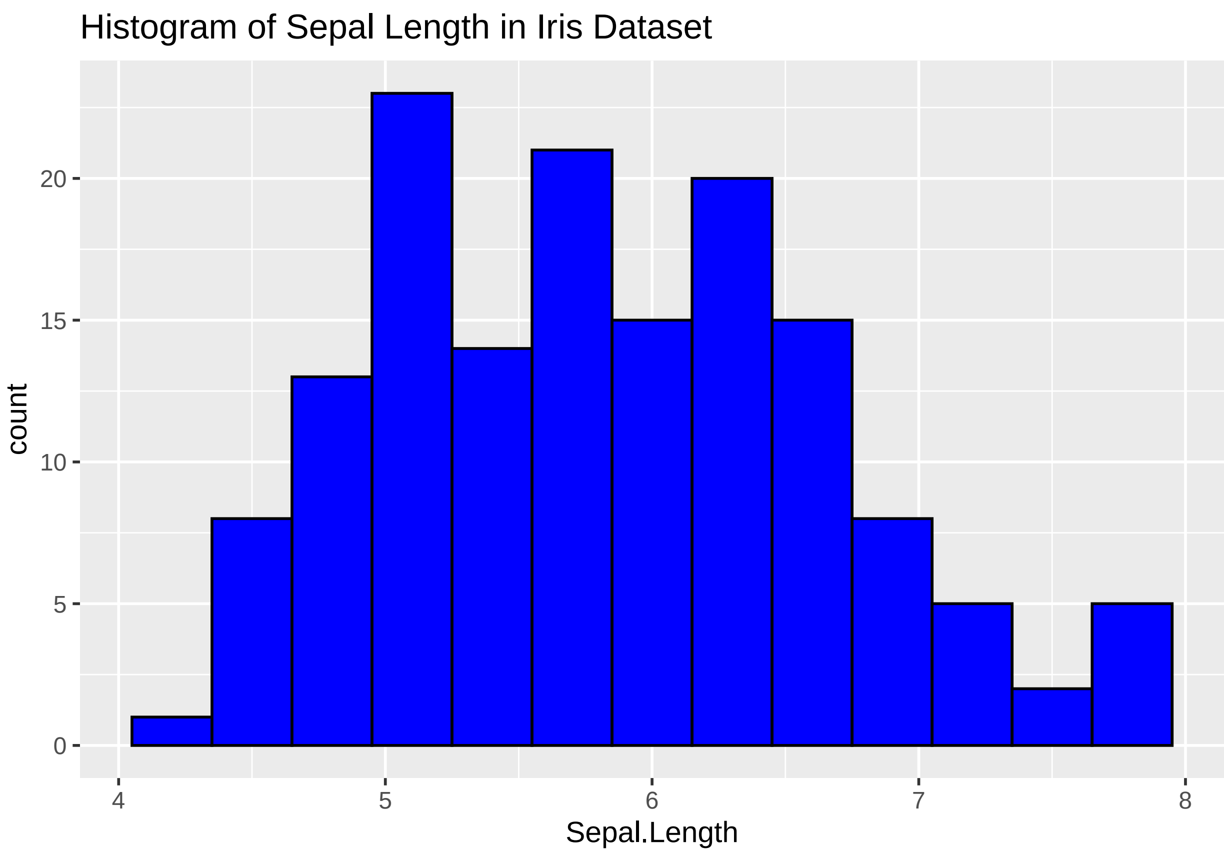
```r
#Skewness
iris_skewness <- sapply(iris[, 1:4], skewness, na.rm = TRUE )
print(paste("Skewness of Iris dataset : ", iris_skewness))
```

```
[1] "Skewness of Iris dataset :  0.311753058502296"
[2] "Skewness of Iris dataset :  0.315767106338938"
[3] "Skewness of Iris dataset :  -0.272127666456721"
[4] "Skewness of Iris dataset :  -0.101934206565599"
```

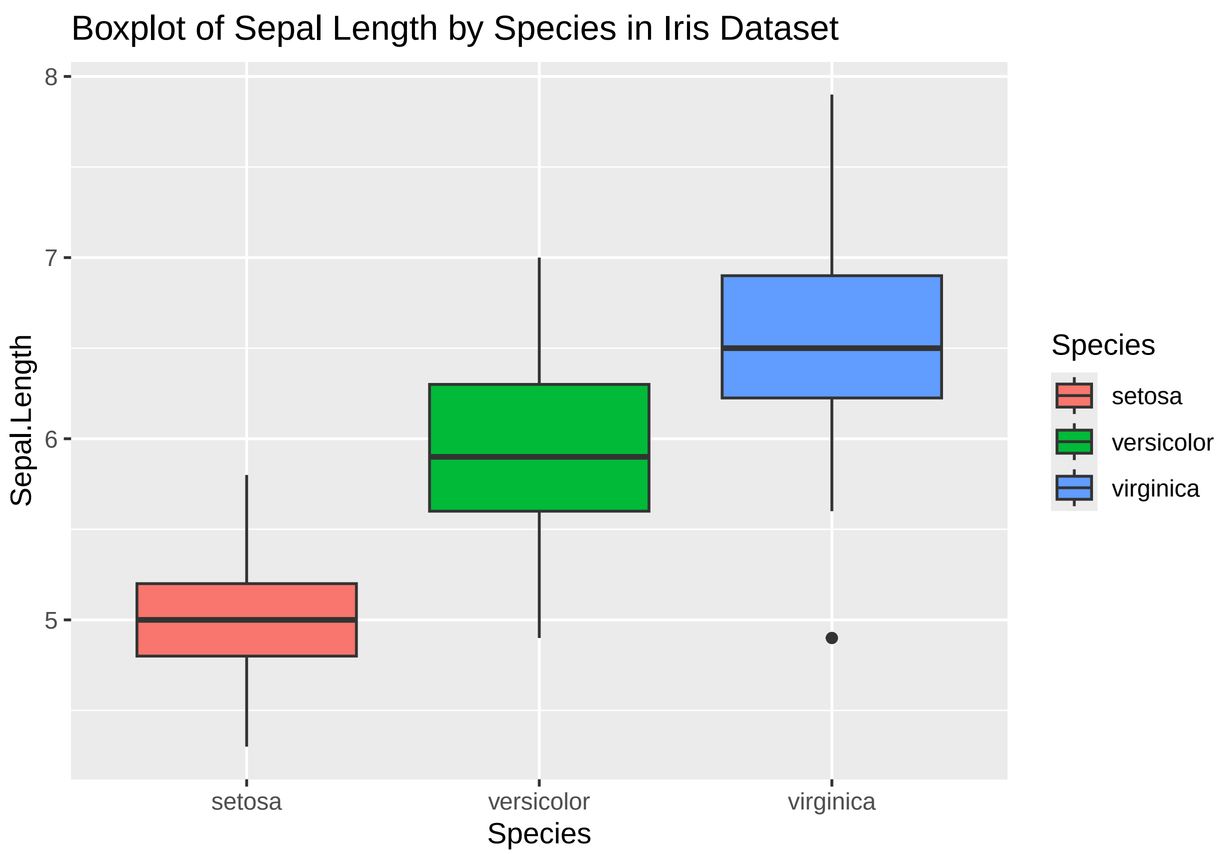
```r
# Hypothesis Testing (t-test) between Sepal.Length of Setosa and Versicolor
setosa <- subset(iris, Species == "setosa")$Sepal.Length
versicolor <- subset(iris, Species == "versicolor")$Sepal.Length
t_test <- t.test(setosa, versicolor)
print(t_test)
```

```
    Welch Two Sample t-test

data:  setosa and versicolor
t = -10.521, df = 86.538, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.1057074 -0.7542926
sample estimates:
mean of x mean of y
    5.006     5.936
```

```
# Visualization of Iris Dataset
# Histogram for Sepal.Length
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram(binwidth = 0.3, fill = "blue", color = "black") +
  ggtitle("Histogram of Sepal Length in Iris Dataset")
```




```
# Boxplot for Sepal.Length across Species
ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_boxplot() +
  ggtitle("Boxplot of Sepal Length by Species in Iris Dataset")
```




```r
print("----- Palmer Penguins Dataset Analysis -----")
```

```
[1] "----- Palmer Penguins Dataset Analysis -----"
```

```r
# Remove rows with missing values
penguins_clean <- na.omit(penguins)

# Mean
penguins_mean <- sapply(penguins_clean[, 3:6], mean, na.rm = TRUE)
print(paste("Mean of Palmer Penguins dataset:", penguins_mean))
```

```
[1] "Mean of Palmer Penguins dataset: 43.9927927927928"
[2] "Mean of Palmer Penguins dataset: 17.1648648648649"
[3] "Mean of Palmer Penguins dataset: 200.966966966967"
[4] "Mean of Palmer Penguins dataset: 4207.05705705706"
```

```r
# Median
penguins_median <- sapply(penguins_clean[, 3:6], median, na.rm = TRUE)
print(paste("Median of Palmer Penguins dataset:", penguins_median))
```

```
[1] "Median of Palmer Penguins dataset: 44.5"
[2] "Median of Palmer Penguins dataset: 17.3"
```

```
[3] "Median of Palmer Penguins dataset: 197"
[4] "Median of Palmer Penguins dataset: 4050"
```

```r
# Mode
penguins_mode <- sapply(penguins_clean[, 3:6], calc_mode)
print(paste("Mode of Palmer Penguins dataset:", penguins_mode))
```

```
[1] "Mode of Palmer Penguins dataset: 41.1"
[2] "Mode of Palmer Penguins dataset: 17"
[3] "Mode of Palmer Penguins dataset: 190"
[4] "Mode of Palmer Penguins dataset: 3800"
```

```r
# Variance
penguins_variance <- sapply(penguins_clean[, 3:6], var, na.rm = TRUE)
print(paste("Variance of Palmer Penguins dataset:", penguins_variance))
```

```
[1] "Variance of Palmer Penguins dataset: 29.9063334418756"
[2] "Variance of Palmer Penguins dataset: 3.87788830999674"
[3] "Variance of Palmer Penguins dataset: 196.441676616375"
[4] "Variance of Palmer Penguins dataset: 648372.487698542"
```

```r
# Standard Deviation
penguins_sd <- sapply(penguins_clean[, 3:6], sd, na.rm = TRUE)
print(paste("Standard Deviation of Palmer Penguins dataset:", penguins_sd))
```

```
[1] "Standard Deviation of Palmer Penguins dataset: 5.46866834264756"
[2] "Standard Deviation of Palmer Penguins dataset: 1.9692354633199"
[3] "Standard Deviation of Palmer Penguins dataset: 14.0157652882879"
[4] "Standard Deviation of Palmer Penguins dataset: 805.215801942897"
```

```r
# Skewness
penguins_skewness <- sapply(penguins_clean[, 3:6], skewness, na.rm = TRUE)
print(paste("Skewness of Palmer Penguins dataset:", penguins_skewness))
```

```
[1] "Skewness of Palmer Penguins dataset: 0.0451359779776739"
[2] "Skewness of Palmer Penguins dataset: -0.149044996398334"
[3] "Skewness of Palmer Penguins dataset: 0.358523654622741"
[4] "Skewness of Palmer Penguins dataset: 0.470116171418382"
```

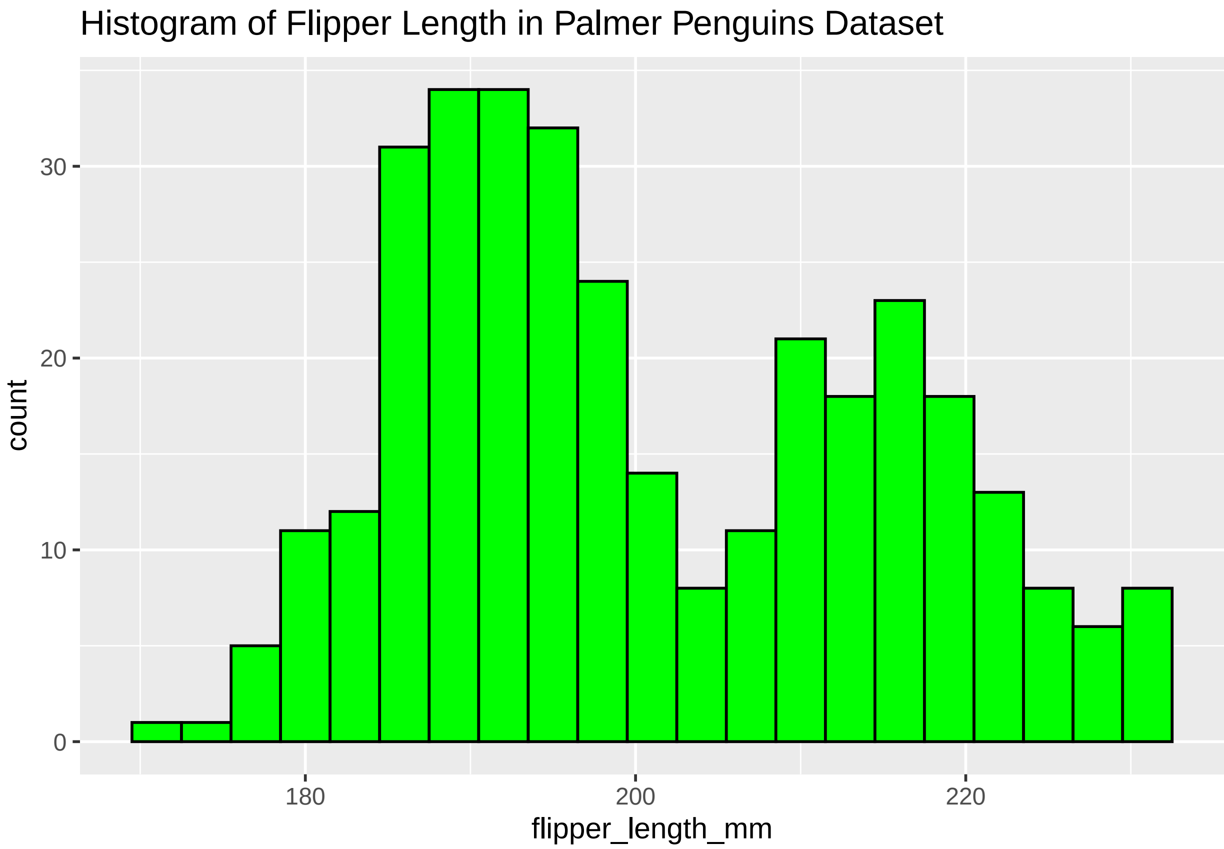
```r
# Kurtosis
penguins_kurtosis <- sapply(penguins_clean[, 3:6], kurtosis, na.rm = TRUE)
print(paste("Kurtosis of Palmer Penguins dataset:", penguins_kurtosis))
```

```
[1] "Kurtosis of Palmer Penguins dataset: 2.11182658541194"
[2] "Kurtosis of Palmer Penguins dataset: 2.10341274887238"
[3] "Kurtosis of Palmer Penguins dataset: 2.03516741259049"
[4] "Kurtosis of Palmer Penguins dataset: 2.25951411974012"
```

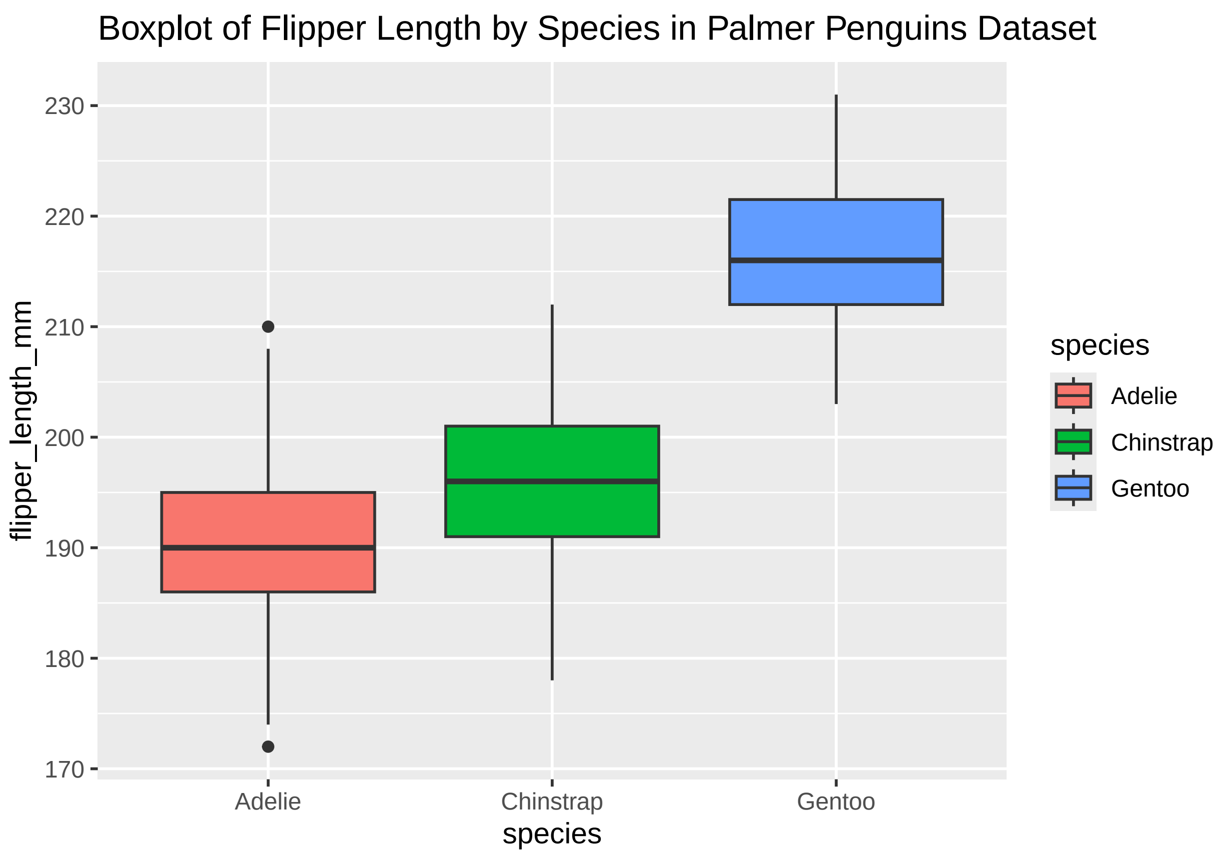
```r
# Hypothesis Testing (t-test) between flipper_length_mm of Adelie and Gentoo species
adelie <- subset(penguins_clean, species == "Adelie")$flipper_length_mm
gentoo <- subset(penguins_clean, species == "Gentoo")$flipper_length_mm
t_test_penguins <- t.test(adelie, gentoo)
print(t_test_penguins)
```

```
    Welch Two Sample t-test

data:  adelie and gentoo
t = -33.506, df = 251.35, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -28.72740 -25.53771
sample estimates:
mean of x mean of y
 190.1027  217.2353
```

```r
# Visualization of Palmer Penguins Dataset
# Histogram for flipper_length_mm
ggplot(penguins_clean, aes(x = flipper_length_mm)) +
  geom_histogram(binwidth = 3, fill = "green", color = "black") +
  ggtitle("Histogram of Flipper Length in Palmer Penguins Dataset")
```




```r
# Boxplot for flipper_length_mm across Species
ggplot(penguins_clean, aes(x = species, y = flipper_length_mm, fill = species)) +
  geom_boxplot() +
  ggtitle("Boxplot of Flipper Length by Species in Palmer Penguins Dataset")
```


Boxplot of Flipper Length by Species in Palmer Penguins Dataset
230
220
210
200
190
180
170
flipper_length_mm
Adelie
Chinstrap
Gentoo
species
species
Adelie
Chinstrap
Gentoo

Program - 4

Data Import, Cleaning, and Export with Titanic Dataset and Adult Income Dataset

Date of Execution - 2025-09-09

Objective: Real World Data Cleaning Processes and emphasis on usage of advanced data wrangling techniques in R.

```r
# Load necessary libraries
library(tidyverse)
library(titanic)
library(dplyr)
library(caret)
library(ggcorrplot)

# Load the Titanic dataset
data <- titanic::titanic_train

# Handle the missing data
# Replace missing values in the 'Age' column with the median age
data$Age[is.na(data$Age)] <- median(data$Age, na.rm = TRUE)

# Replace missing values in the 'Embarked' column with the mode
mode_embarked <- as.character(names(sort(table(data$Embarked), decreasing = TRUE)[1]))
data$Embarked[is.na(data$Embarked)] <- mode_embarked

# Define the numeric columns for z-score and correlation calculation
numeric_columns <- c("Age", "SibSp", "Parch", "Fare", "Survived", "Pclass")

# Remove outliers using z-score
z_scores <- as.data.frame(scale(data[, numeric_columns]))

# Identify the rows that have z_scores greater than 3 or less than -3 (outliers)
outlier_rows <- apply(z_scores, 1, function(row) any(abs(row) > 3))

# Filter out Outliers
data_cleaned <- data[!outlier_rows, ]
```

```r
# Summarize the dataset before and after cleaning
summary_before <- summary(data)
summary_after <- summary(data_cleaned)

# Calculate Correlation Matrix (fixed)
correlation_matrix <- cor(data_cleaned[, numeric_columns], use = "complete.obs")

# Export cleaned data onto a new CSV file
write.csv(data_cleaned, "titanic_cleaned.csv", row.names = FALSE)

# Display Summaries
print("Summary Before Cleaning:")
```

```
[1] "Summary Before Cleaning:"
```

```r
print(summary_before)
```

```
  PassengerId       Survived          Pclass          Name          
 Min.   :  1.0   Min.   :0.0000   Min.   :1.000   Length:891        
 1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000   Class :character  
 Median :446.0   Median :0.0000   Median :3.000   Mode  :character  
 Mean   :446.0   Mean   :0.3838   Mean   :2.309                     
 3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000                     
 Max.   :891.0   Max.   :1.0000   Max.   :3.000                     
     Sex                 Age            SibSp           Parch       
 Length:891         Min.   : 0.42   Min.   :0.000   Min.   :0.0000  
 Class :character   1st Qu.:22.00   1st Qu.:0.000   1st Qu.:0.0000  
 Mode  :character   Median :28.00   Median :0.000   Median :0.0000  
                    Mean   :29.36   Mean   :0.523   Mean   :0.3816  
                    3rd Qu.:35.00   3rd Qu.:1.000   3rd Qu.:0.0000  
                    Max.   :80.00   Max.   :8.000   Max.   :6.0000  
    Ticket               Fare           Cabin          
 Length:891         Min.   :  0.00   Length:891        
 Class :character   1st Qu.:  7.91   Class :character  
 Mode  :character   Median : 14.45   Mode  :character  
                    Mean   : 32.20                     
                    3rd Qu.: 31.00                     
                    Max.   :512.33                     
   Embarked        
 Length:891        
 Class :character  
```

```
 Mode  :character
```

```r
print("Summary After Cleaning:")
```

```
[1] "Summary After Cleaning:"
```

```r
print(summary_after)
```

```
  PassengerId       Survived          Pclass          Name          
 Min.   :  1.0   Min.   :0.0000   Min.   :1.000   Length:820        
 1st Qu.:226.8   1st Qu.:0.0000   1st Qu.:2.000   Class :character  
 Median :446.5   Median :0.0000   Median :3.000   Mode  :character  
 Mean   :445.7   Mean   :0.3902   Mean   :2.311                     
 3rd Qu.:661.2   3rd Qu.:1.0000   3rd Qu.:3.000                     
 Max.   :891.0   Max.   :1.0000   Max.   :3.000                     
     Sex                  Age            SibSp            Parch       
 Length:820         Min.   : 0.42   Min.   :0.0000   Min.   :0.0000  
 Class :character   1st Qu.:23.00   1st Qu.:0.0000   1st Qu.:0.0000  
 Mode  :character   Median :28.00   Median :0.0000   Median :0.0000  
                    Mean   :29.44   Mean   :0.3488   Mean   :0.2549  
                    3rd Qu.:35.00   3rd Qu.:1.0000   3rd Qu.:0.0000  
                    Max.   :66.00   Max.   :3.0000   Max.   :2.0000  
    Ticket               Fare            Cabin          
 Length:820         Min.   :  0.000   Length:820        
 Class :character   1st Qu.:  7.896   Class :character  
 Mode  :character   Median : 13.000   Mode  :character  
                    Mean   : 25.836                     
                    3rd Qu.: 27.000                     
                    Max.   :164.867                     
   Embarked        
 Length:820        
 Class :character  
 Mode  :character  
```

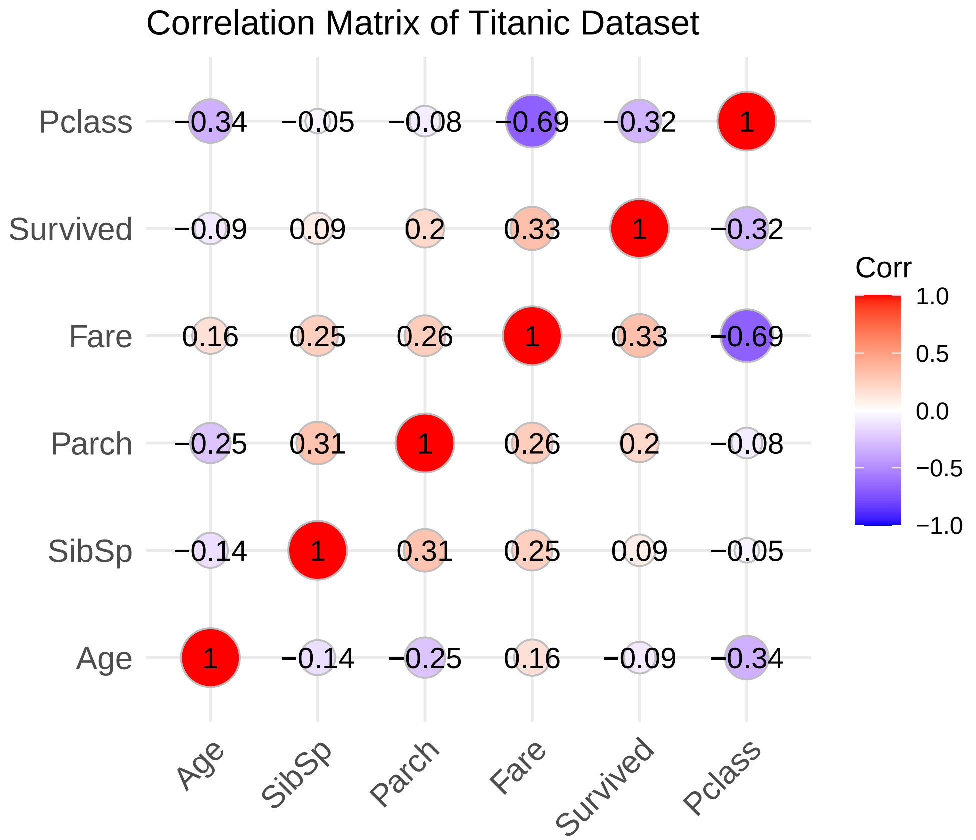
```r
print("Correlation Matrix:")
```

```
[1] "Correlation Matrix:"
```

```r
print(correlation_matrix)
```

```
                 Age       SibSp      Parch       Fare    Survived
Age       1.00000000 -0.14391182 -0.2517719  0.1598100 -0.08602643
SibSp    -0.14391182  1.00000000  0.3072105  0.2472157  0.09445934
Parch    -0.25177192  0.30721046  1.0000000  0.2599031  0.20107069
Fare      0.15981001  0.24721568  0.2599031  1.0000000  0.33043946
Survived -0.08602643  0.09445934  0.2010707  0.3304395  1.00000000
Pclass   -0.33698055 -0.05231213 -0.0783660 -0.6917198 -0.32230582
              Pclass
Age      -0.33698055
SibSp    -0.05231213
Parch    -0.07836600
Fare     -0.69171982
Survived -0.32230582
Pclass    1.00000000
```

```r
# Visualize Correlation Matrix (fixed)
ggcorrplot(correlation_matrix,
           method = "circle",
           lab = TRUE) +
  ggtitle("Correlation Matrix of Titanic Dataset")
```



Objective - Data Import, Cleaning, and Export with Adult Income Dataset

```r
# Load necessary libraries
library(tidyverse)
library(dplyr)
library(caret)
library(ggcorrplot)

# Load the Adult Income dataset
data <- read.csv("D:/Coding/Coding/Time Series Analysis/Lab 4/adult.data", header = FALSE)

# Assign column names based on the dataset documentation
colnames(data) <- c('age', 'workclass', 'fnlwgt', 'education', 'education_num',
                    'marital_status', 'occupation', 'relationship', 'race', 'sex', 'capital_gain',
                    'capital_loss', 'hours_per_week', 'native_country', 'income')

# Handle missing values represented by '?'
data[data == '?'] <- NA

# Replace categorical missing values with mode
replace_mode <- function(x){
  mode_val <- as.character(names(sort(table(x), decreasing = TRUE)[1]))
```

```
  x[is.na(x)] <- mode_val
  return(x)
}

data <- data %>%
  mutate_if(is.character, replace_mode)

# Replace numeric missing values with median
data <- data %>%
  mutate_if(is.numeric, ~ifelse(is.na(.), median(., na.rm = TRUE), .))

# Define the remove_outliers function
remove_outliers <- function(x){
  z_scores <- scale(x)
  x[abs(z_scores) <= 3]
}

# Remove outliers using z-score
numeric_columns <- sapply(data, is.numeric)

# Apply z-score outlier removal to numeric columns
data_cleaned <- data %>%
  filter(!apply(as.data.frame(scale(data[, numeric_columns])), 1,
                function(row) any(abs(row) > 3)))

# Summarize before and after cleaning
summary_before <- summary(data)
summary_after <- summary(data_cleaned)

# Calculate correlation Matrix
correlation_matrix <- cor(data_cleaned[, numeric_columns], use = "complete.obs")

# Export as CSV
write.csv(data_cleaned, "cleaned_adult_income_data.csv", row.names = FALSE)

# Display Summaries
print("Summary Before Cleaning:")
```

```
[1] "Summary Before Cleaning:"
```

```r
print(summary_before)
```

```
      age         workclass             fnlwgt       
 Min.   :17.00   Length:32561       Min.   :  12285  
 1st Qu.:28.00   Class :character   1st Qu.: 117827  
 Median :37.00   Mode  :character   Median : 178356  
 Mean   :38.58                      Mean   : 189778  
 3rd Qu.:48.00                      3rd Qu.: 237051  
 Max.   :90.00                      Max.   :1484705  
  education         education_num   marital_status    
 Length:32561       Min.   : 1.00   Length:32561      
 Class :character   1st Qu.: 9.00   Class :character  
 Mode  :character   Median :10.00   Mode  :character  
                    Mean   :10.08                     
                    3rd Qu.:12.00                     
                    Max.   :16.00                     
  occupation        relationship           race          
 Length:32561       Length:32561       Length:32561      
 Class :character   Class :character   Class :character  
 Mode  :character   Mode  :character   Mode  :character  
                                                         
                                                         
                                                         
     sex             capital_gain    capital_loss    hours_per_week 
 Length:32561       Min.   :    0   Min.   :   0.0   Min.   : 1.00  
 Class :character   1st Qu.:    0   1st Qu.:   0.0   1st Qu.:40.00  
 Mode  :character   Median :    0   Median :   0.0   Median :40.00  
                    Mean   : 1078   Mean   :  87.3   Mean   :40.44  
                    3rd Qu.:    0   3rd Qu.:   0.0   3rd Qu.:45.00  
                    Max.   :99999   Max.   :4356.0   Max.   :99.00  
 native_country        income         
 Length:32561       Length:32561      
 Class :character   Class :character  
 Mode  :character   Mode  :character  
```

```r
print("Summary After Cleaning:")
```

```
[1] "Summary After Cleaning:"
```

```r
print(summary_after)
```

```
      age          workclass             fnlwgt      
 Min.   :17.00   Length:29828       Min.   : 12285  
 1st Qu.:27.00   Class :character   1st Qu.:117509  
 Median :37.00   Mode  :character   Median :177667  
 Mean   :38.14                      Mean   :185193  
 3rd Qu.:47.00                      3rd Qu.:234279  
 Max.   :79.00                      Max.   :506329  
  education          education_num   marital_status    
 Length:29828       Min.   : 3.00   Length:29828      
 Class :character   1st Qu.: 9.00   Class :character  
 Mode  :character   Median :10.00   Mode  :character  
                    Mean   :10.08                     
                    3rd Qu.:12.00                     
                    Max.   :16.00                     
  occupation         relationship           race          
 Length:29828       Length:29828       Length:29828      
 Class :character   Class :character   Class :character  
 Mode  :character   Mode  :character   Mode  :character  
```

```
     sex             capital_gain      capital_loss    
 Length:29828       Min.   :    0.0   Min.   :   0.000  
 Class :character   1st Qu.:    0.0   1st Qu.:   0.000  
 Mode  :character   Median :    0.0   Median :   0.000  
                    Mean   :  570.2   Mean   :   1.209  
                    3rd Qu.:    0.0   3rd Qu.:   0.000  
                    Max.   :22040.0   Max.   :1258.000  
 hours_per_week native_country        income         
 Min.   : 4.0   Length:29828       Length:29828      
 1st Qu.:40.0   Class :character   Class :character  
 Median :40.0   Mode  :character   Mode  :character  
 Mean   :39.9                                        
 3rd Qu.:45.0                                        
 Max.   :77.0                                        
```

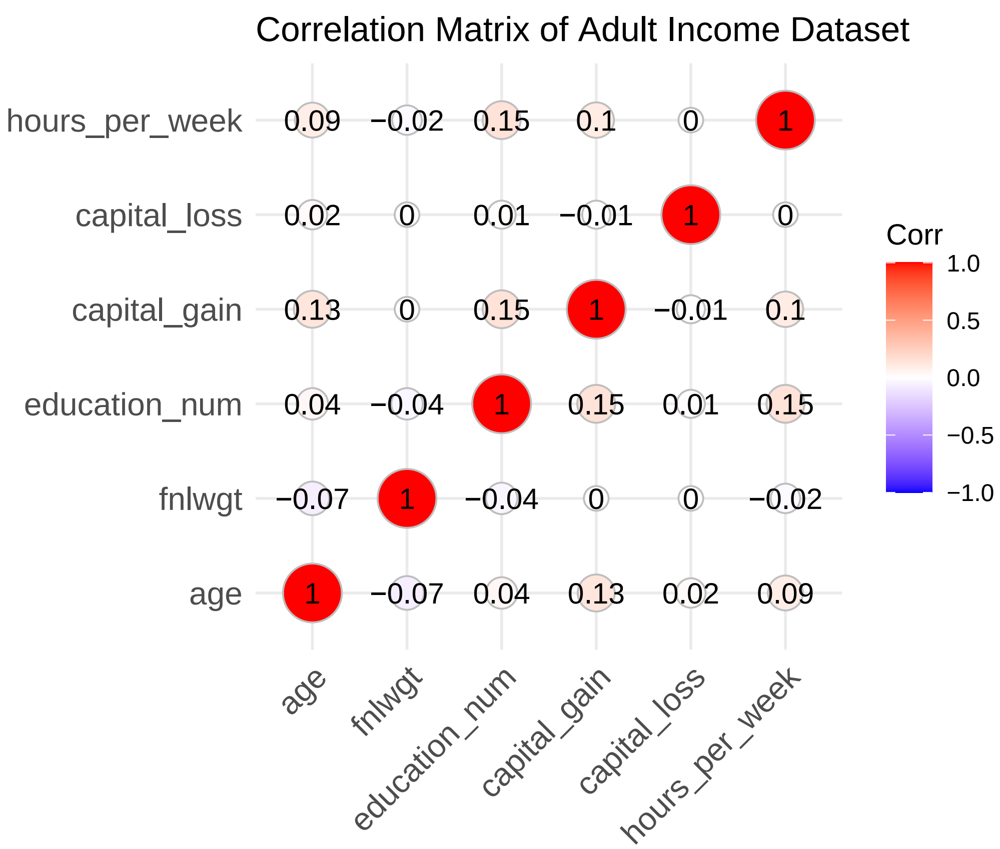
```r
print("Correlation Matrix:")
```

```
[1] "Correlation Matrix:"
```

```r
print(correlation_matrix)
```

```
                      age       fnlwgt education_num capital_gain
age             1.00000000 -0.074427786   0.041427102  0.131043981
fnlwgt         -0.07442779  1.000000000  -0.037482414 -0.002378925
education_num   0.04142710 -0.037482414   1.000000000  0.154844283
capital_gain    0.13104398 -0.002378925   0.154844283  1.000000000
capital_loss    0.02082465  0.002583047   0.009481359 -0.009038231
hours_per_week  0.09219535 -0.015375555   0.150513483  0.097209049
               capital_loss hours_per_week
age             0.020824647    0.092195352
fnlwgt          0.002583047   -0.015375555
education_num   0.009481359    0.150513483
capital_gain   -0.009038231    0.097209049
capital_loss    1.000000000   -0.003089539
hours_per_week -0.003089539    1.000000000
```

```r
# Visualize Correlation Matrix (fixed)
ggcorrplot(correlation_matrix,
           method = "circle",
           lab = TRUE) +
  ggtitle("Correlation Matrix of Adult Income Dataset")
```



Program - 5

Advanced Data Manipulation with dplyr and Complex Grouping

Date of Execution - 2025-09-16

Objective - The goal of this program is to test advanced data manipulation techniques using the dplyr package.

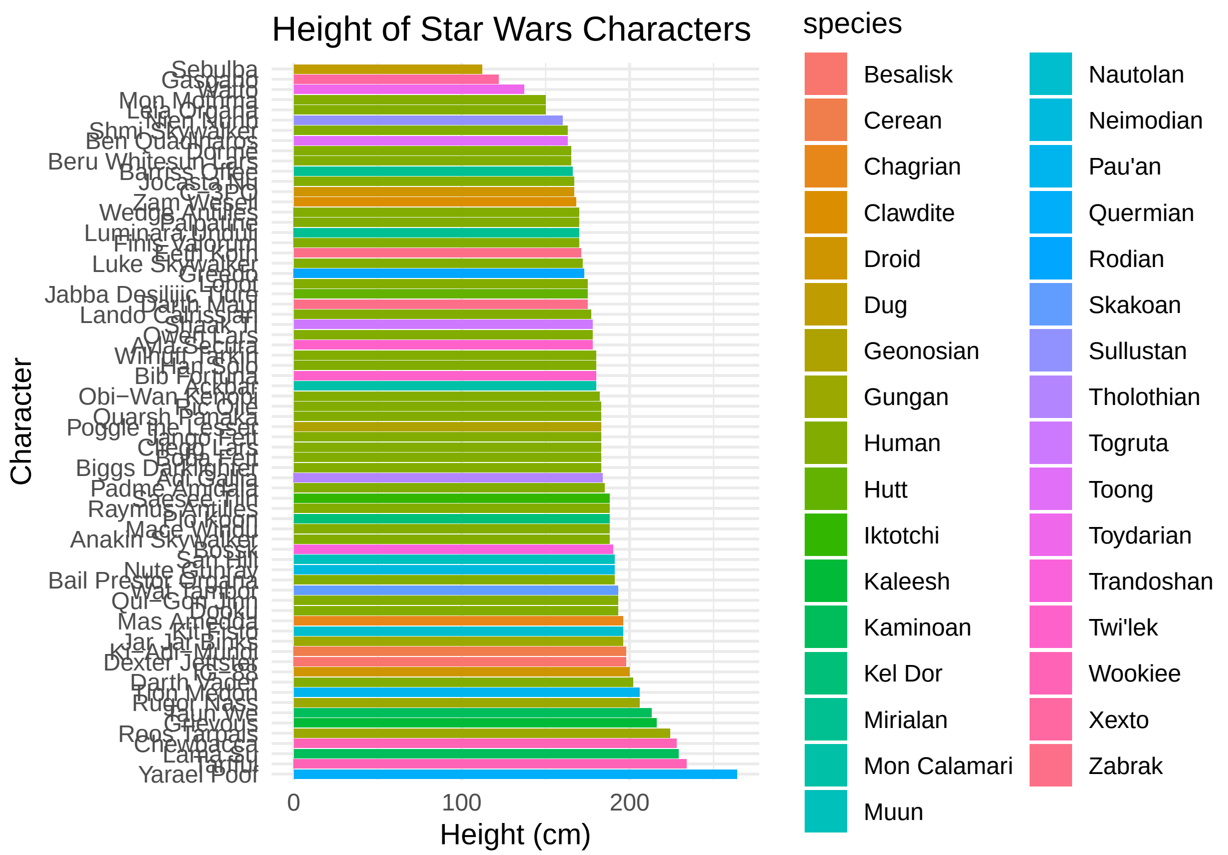
```r
## Load necessary libraries
library(dplyr)
library(nycflights13)
library(ggplot2)
library(zoo)

# Preview the Star Wars Dataset
data("starwars")
head(starwars)
```

```
# A tibble: 6 x 14
  name   height  mass hair_color skin_color eye_color birth_year sex
  <chr>   <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
1 Luke ~    172    77 blond      fair       blue            19   male
2 C-3PO     167    75 <NA>       gold       yellow         112   none
3 R2-D2      96    32 <NA>       white, bl~ red             33   none
4 Darth~    202   136 none       white      yellow          41.9 male
5 Leia ~    150    49 brown      light      brown           19   fema~
6 Owen ~    178   120 brown, gr~ light      blue            52   male
# i 6 more variables: gender <chr>, homeworld <chr>, species <chr>,
#   films <list>, vehicles <list>, starships <list>
```

```r
# Select specific columns (name, species, height, mass),
# filter out rows with missing species or height,
# and arrange by height in descending order
starwars_filtered <- starwars %>%
  select(name, species, height, mass) %>%
  filter(!is.na(species) & !is.na(height) & height > 100) %>%
  arrange(desc(height))

# Display the filtered data
head(starwars_filtered)
```

```
# A tibble: 6 x 4
  name         species  height  mass
  <chr>        <chr>     <int> <dbl>
1 Yarael Poof  Quermian    264    NA
2 Tarfful      Wookiee     234   136
3 Lama Su      Kaminoan    229    88
4 Chewbacca    Wookiee     228   112
5 Roos Tarpals Gungan      224    82
6 Grievous     Kaleesh     216   159
```

```r
# Plotting the filtered data
ggplot(starwars_filtered, aes(x = reorder(name, -height), y = height, fill = species)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Height of Star Wars Characters",
       x = "Character",
       y = "Height (cm)") +
  theme_minimal()
```

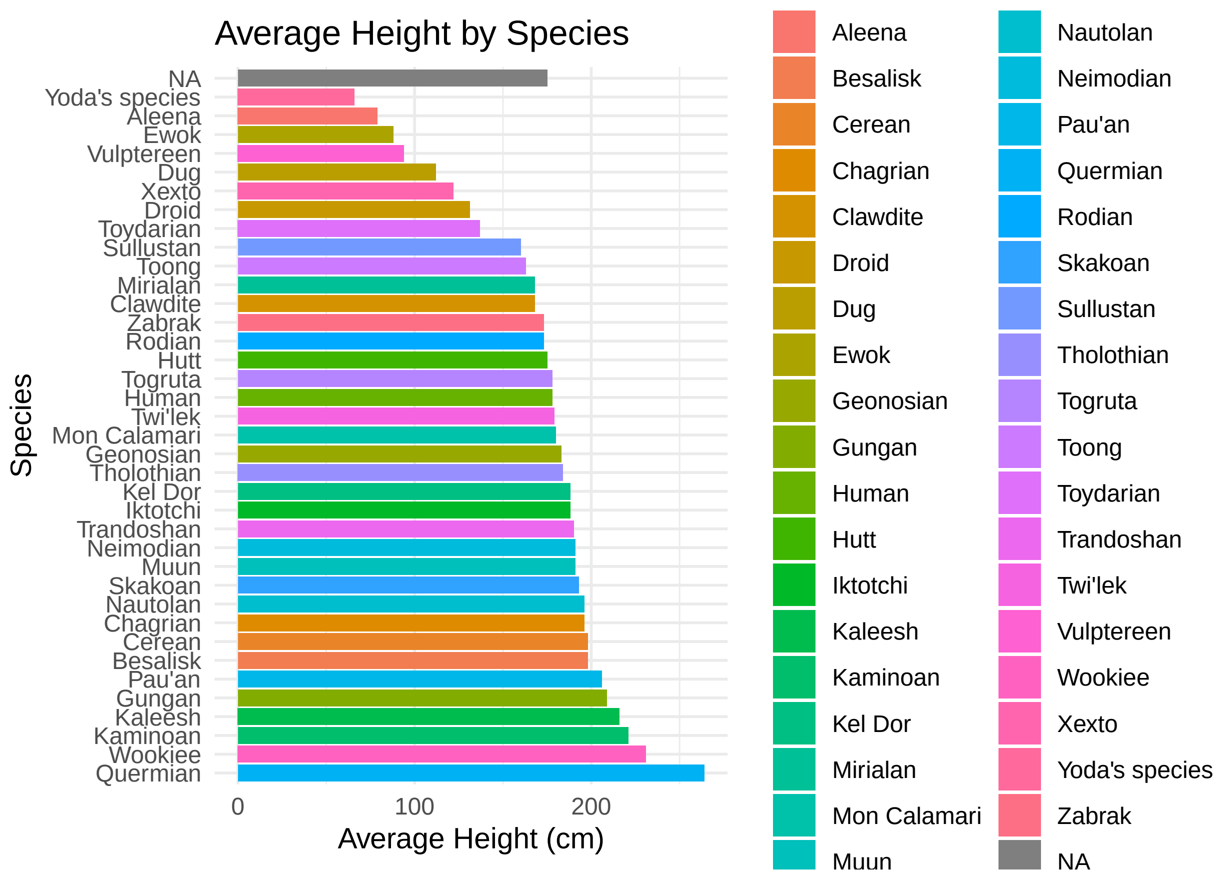



```r
# Grouping by species, calculating average height and mass, and counting observation
species_summary <- starwars %>%
  group_by(species) %>%
```

```r
  summarise(
    avg_height = mean(height, na.rm = TRUE),
    avg_mass = mean(mass, na.rm = TRUE),
    count = n()
  ) %>%
  arrange(desc(count))

# Display the species summary
head(species_summary)
```

```
# A tibble: 6 x 4
  species  avg_height avg_mass count
  <chr>         <dbl>    <dbl> <int>
1 Human          178      81.3    35
2 Droid          131.     69.8     6
3 <NA>           175      81       4
4 Gungan         209.     74       3
5 Kaminoan       221      88       2
6 Mirialan       168      53.1     2
```

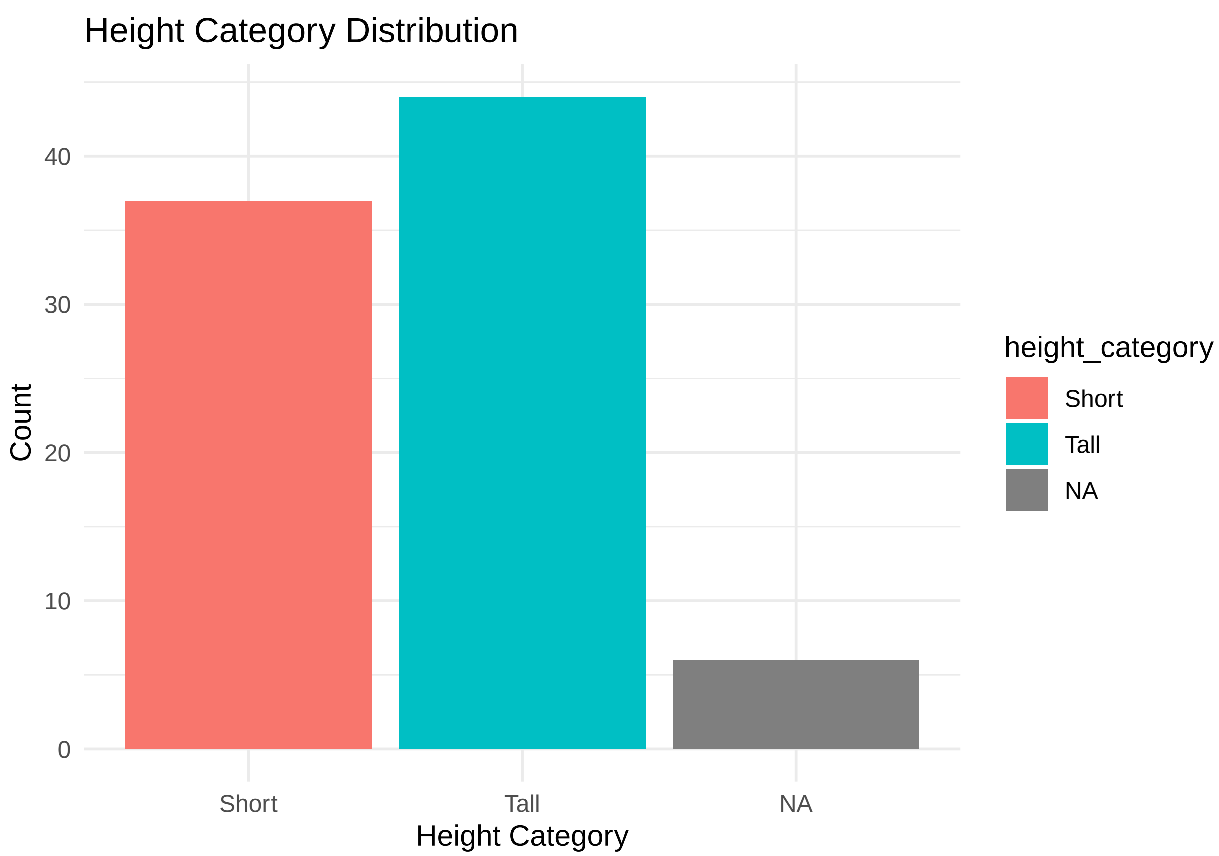
```r
# Plotting the average height
ggplot(species_summary, aes(x = reorder(species, -avg_height), y = avg_height, fill = species)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Average Height by Species",
       x = "Species",
       y = "Average Height (cm)") +
  theme_minimal()
```




```r
# Adding a new column that classifies characters based on height
starwars_classified <- starwars %>%
  mutate(height_category = ifelse(height < 180, "Short", "Tall"))

# Display the classified data
head(starwars_classified)
```

```
# A tibble: 6 x 15
  name   height  mass hair_color skin_color eye_color birth_year sex
  <chr>   <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
1 Luke ~    172    77 blond      fair       blue            19   male
2 C-3PO     167    75 <NA>       gold       yellow         112   none
3 R2-D2      96    32 <NA>       white, bl~ red             33   none
4 Darth~    202   136 none       white      yellow          41.9 male
5 Leia ~    150    49 brown      light      brown           19   fema~
6 Owen ~    178   120 brown, gr~ light      blue            52   male
# i 7 more variables: gender <chr>, homeworld <chr>, species <chr>,
#   films <list>, vehicles <list>, starships <list>,
#   height_category <chr>
```

```r
# Plotting height Category distribution
ggplot(starwars_classified, aes(x = height_category, fill = height_category)) +
  geom_bar() +
  labs(title = "Height Category Distribution",
       x = "Height Category",
       y = "Count") +
  theme_minimal()
```




```r
# Joining with another dataset (flights dataset from nycflights13)
data("flights")
data("airlines")

# Inner join flights with airlines on the common column "carrier"
flights_inner_join <- flights %>%
  inner_join(airlines, by = "carrier")

# Outer join flights with airlines on the common column "carrier"
flights_outer_join <- flights %>%
  full_join(airlines, by = "carrier")

# Display the joined data
head(flights_inner_join)
```

```
# A tibble: 6 x 20
   year month   day dep_time sched_dep_time dep_delay arr_time
  <int> <int> <int>    <int>          <int>     <dbl>    <int>
1  2013     1     1      517            515         2      830
2  2013     1     1      533            529         4      850
3  2013     1     1      542            540         2      923
4  2013     1     1      544            545        -1     1004
5  2013     1     1      554            600        -6      812
6  2013     1     1      554            558        -4      740
# i 13 more variables: sched_arr_time <int>, arr_delay <dbl>,
#   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>,
#   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#   minute <dbl>, time_hour <dttm>, name <chr>
```

```r
head(flights_outer_join)
```

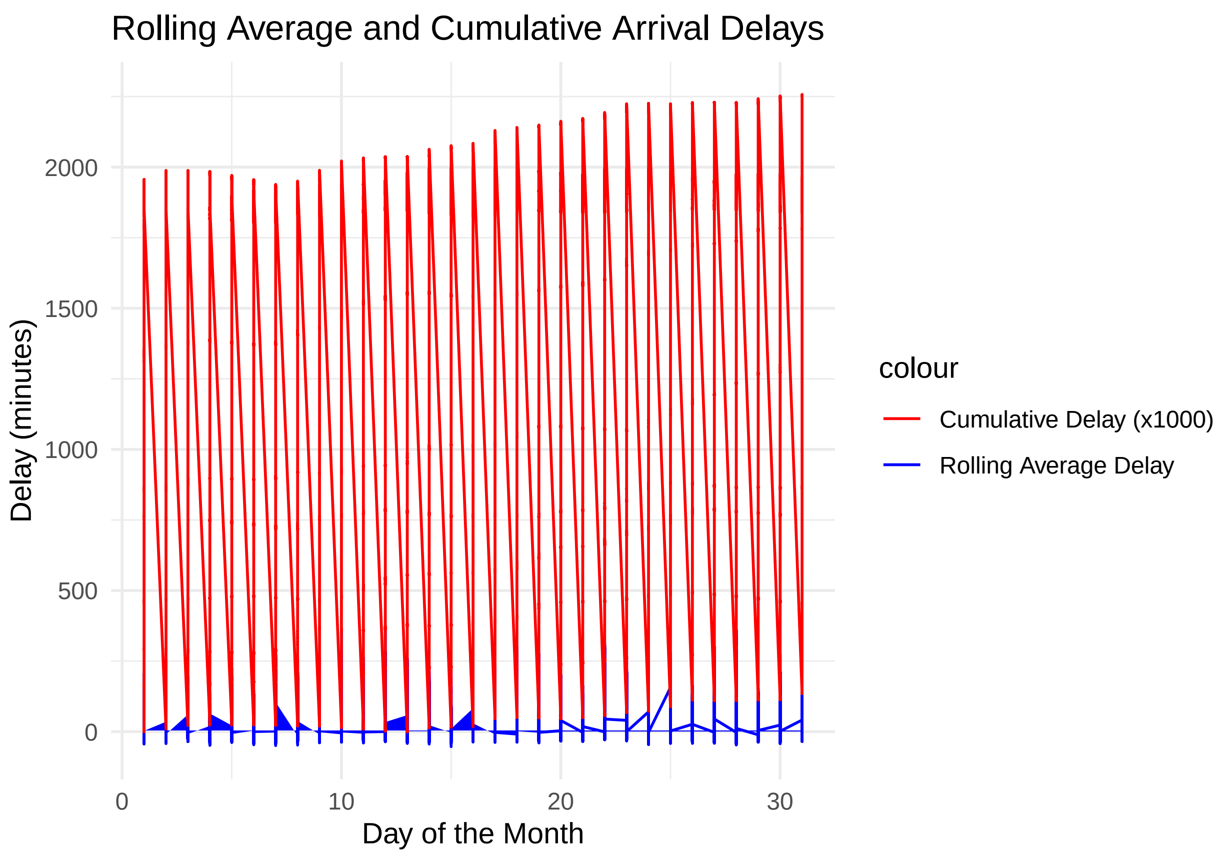
```
# A tibble: 6 x 20
   year month   day dep_time sched_dep_time dep_delay arr_time
  <int> <int> <int>    <int>          <int>     <dbl>    <int>
1  2013     1     1      517            515         2      830
2  2013     1     1      533            529         4      850
3  2013     1     1      542            540         2      923
4  2013     1     1      544            545        -1     1004
5  2013     1     1      554            600        -6      812
6  2013     1     1      554            558        -4      740
# i 13 more variables: sched_arr_time <int>, arr_delay <dbl>,
#   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>,
#   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#   minute <dbl>, time_hour <dttm>, name <chr>
```

```r
# Calculating a 5 period rolling average of arrival delays and cumulative sum
flights_rolling <- flights %>%
  arrange(year, month, day) %>%
  mutate(
    arr_delay = ifelse(is.na(arr_delay), 0, arr_delay),
    rolling_avg_delay = zoo::rollmean(arr_delay, 5, fill = NA),
    cumulative_delay = cumsum(arr_delay)
  )

# Display the transformed data
head(flights_rolling)
```

```
# A tibble: 6 x 21
   year month   day dep_time sched_dep_time dep_delay arr_time
  <int> <int> <int>    <int>          <int>     <dbl>    <int>
1  2013     1     1      517            515         2      830
2  2013     1     1      533            529         4      850
3  2013     1     1      542            540         2      923
4  2013     1     1      544            545        -1     1004
5  2013     1     1      554            600        -6      812
6  2013     1     1      554            558        -4      740
# i 14 more variables: sched_arr_time <int>, arr_delay <dbl>,
#   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>,
#   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#   minute <dbl>, time_hour <dttm>, rolling_avg_delay <dbl>,
#   cumulative_delay <dbl>
```

```
# Plotting the rolling average and cumulative delays
ggplot(flights_rolling, aes(x = day)) +
  geom_line(aes(y = rolling_avg_delay, color = "Rolling Average Delay")) +
  geom_line(aes(y = cumulative_delay / 1000, color = "Cumulative Delay (x1000)")) +
  labs(title = "Rolling Average and Cumulative Arrival Delays",
       x = "Day of the Month",
       y = "Delay (minutes)") +
  scale_color_manual(values = c("Rolling Average Delay" = "blue",
                                "Cumulative Delay (x1000)" = "red")) +
  theme_minimal()
```

```
Warning: Removed 4 rows containing missing values or values outside the scale
range (‘geom_line()‘).
```


Rolling Average and Cumulative Arrival Delays
Delay (minutes)
2000
1500
1000
500
0
0
10
20
30
Day of the Month
colour
Cumulative Delay (x1000)
Rolling Average Delay

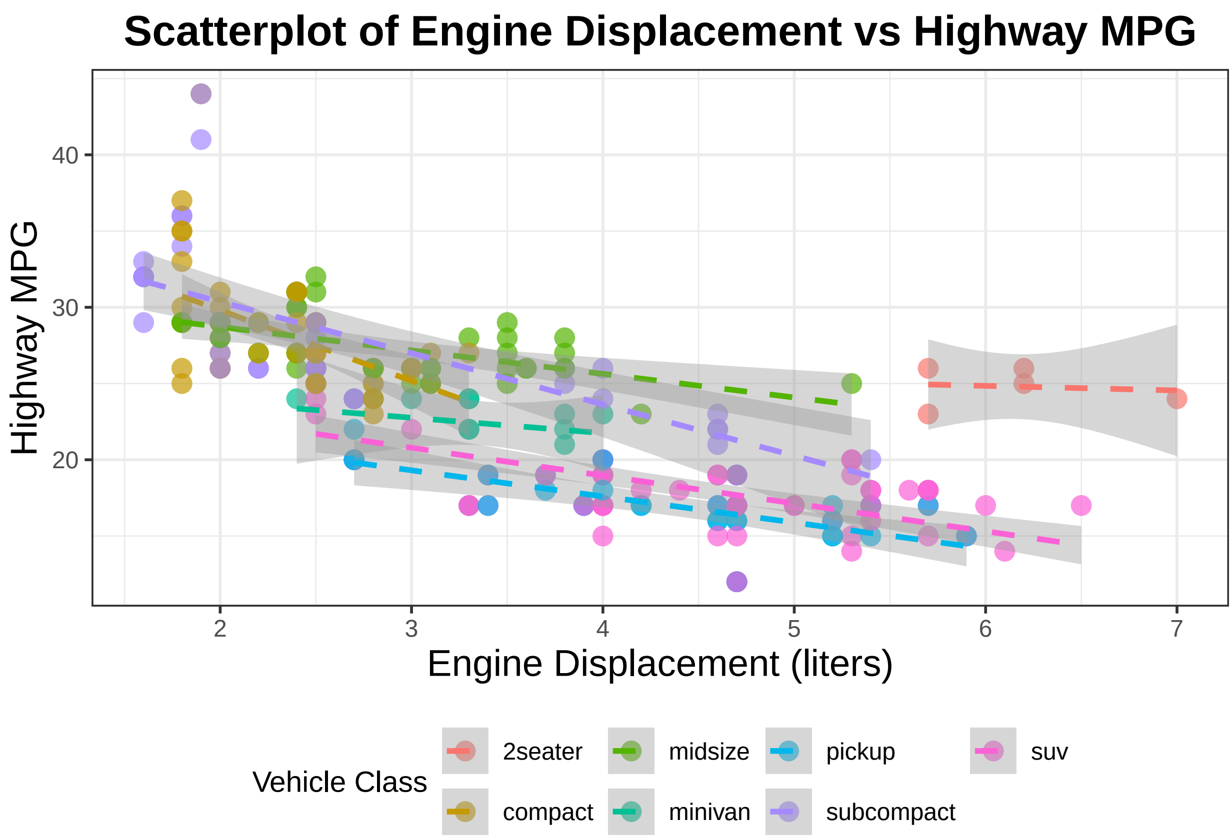
Program - 6

Data Visualisation with ggplot2 and Customisations

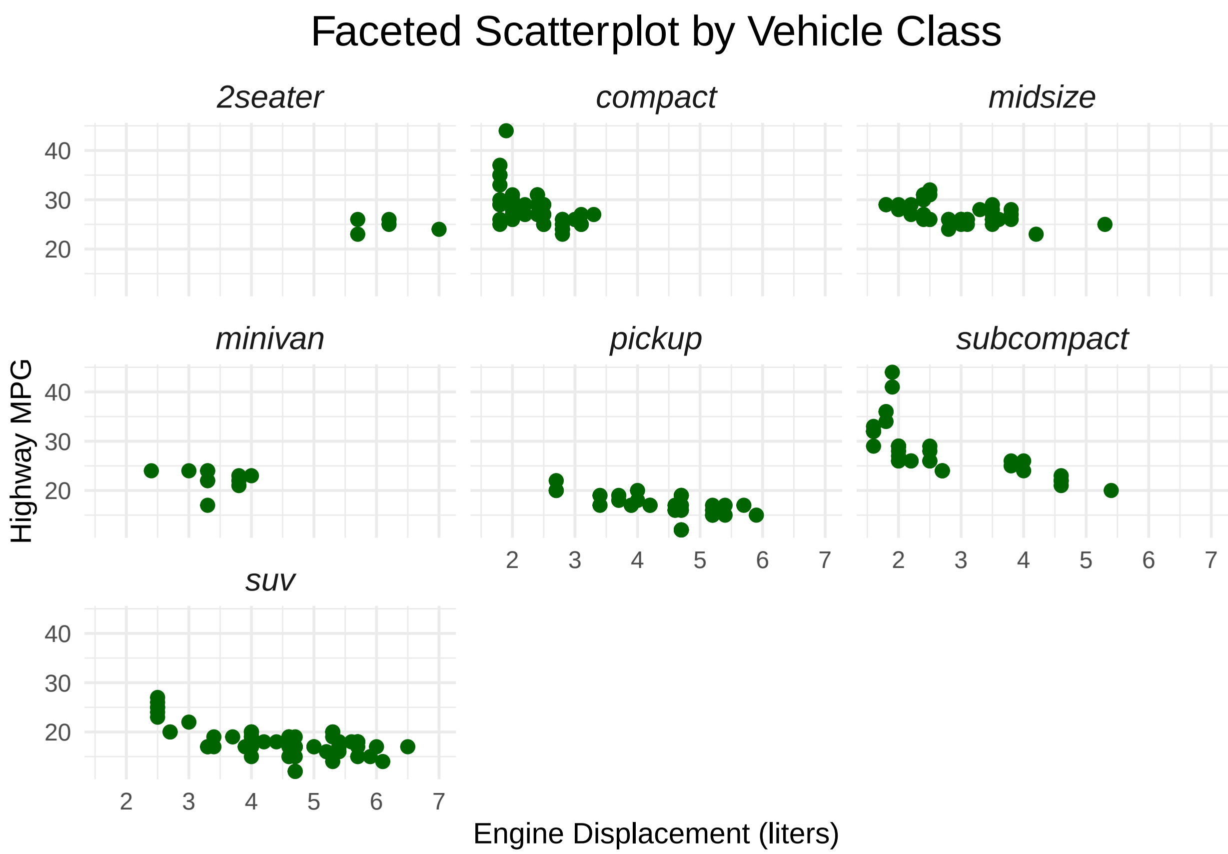
Date of Execution - 2025-09-23

Objective - This program evaluates students' ability to create and customize complex data visualizations using the ggplot2 package.

```r
# Load necessary libraries
library(ggplot2)
library(dplyr)
library(reshape2)

# Scatterplot with regression line and confidence intervals
data("mpg")

ggplot(mpg, aes(x = displ, y = hwy, color = class)) +
  geom_point(size = 3, alpha = 0.7) +
  geom_smooth(method = "lm", se = TRUE, linetype = "dashed") +
  labs(title = "Scatterplot of Engine Displacement vs Highway MPG",
       x = "Engine Displacement (liters)",
       y = "Highway MPG",
       color = "Vehicle Class") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
        axis.title = element_text(size = 14),
        legend.position = "bottom")
```

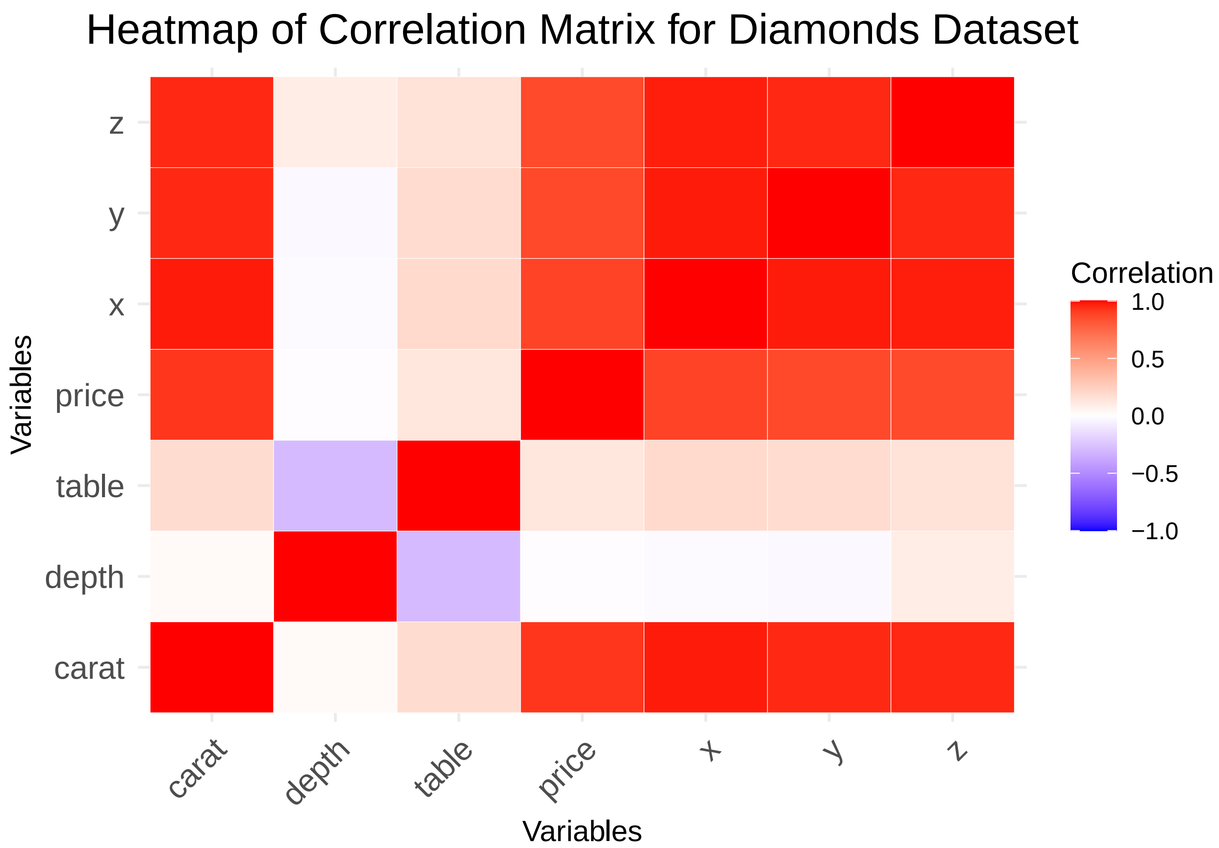



```
# Multi-panel plot using Faceting
# Creating faceted scatter plots by vehicle class with enhanced aesthetics
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point(color = "darkgreen", size = 2) +
  facet_wrap(~ class, ncol = 3) +
  labs(title = "Faceted Scatterplot by Vehicle Class",
       x = "Engine Displacement (liters)",
       y = "Highway MPG",
       color = "Drive Type") +
  theme_minimal() +
  theme(strip.text = element_text(size = 12, face = "italic"),
        plot.title = element_text(hjust = 0.5, size = 16))
```

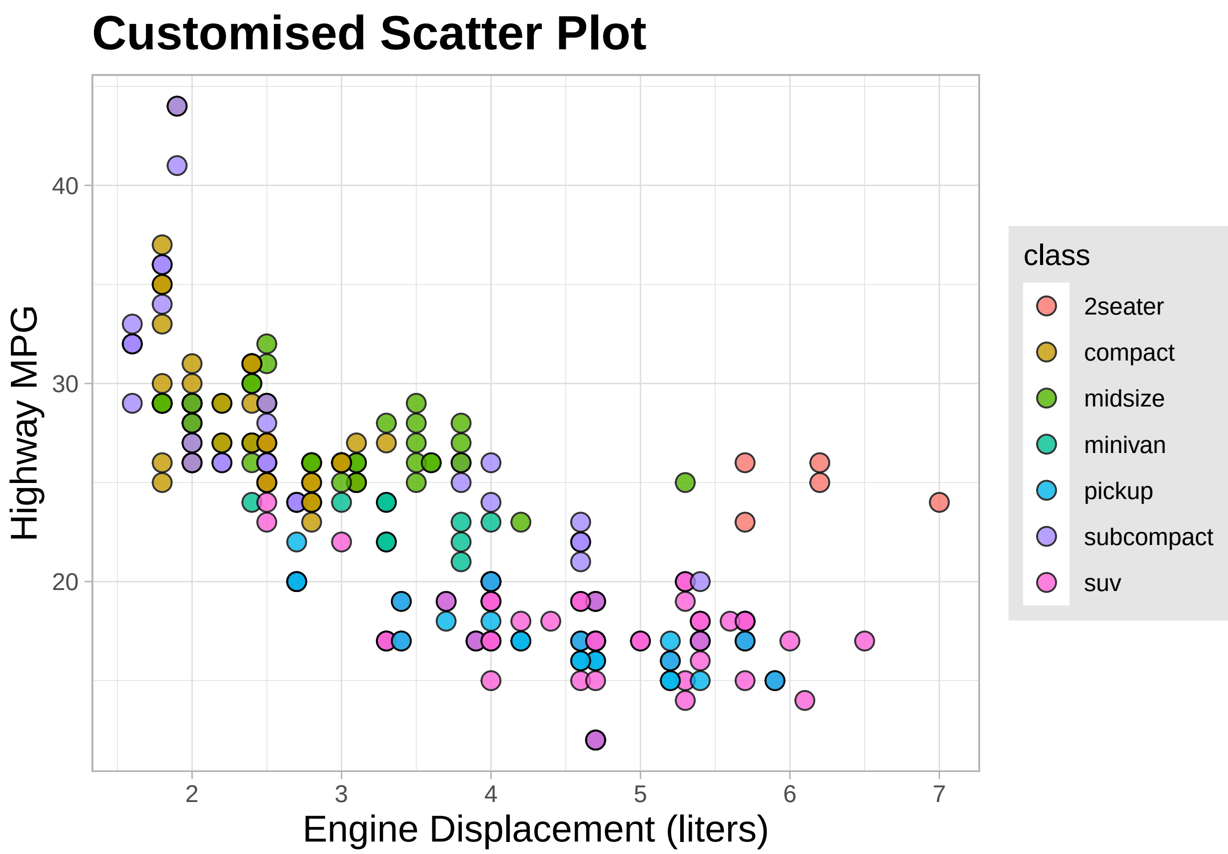



```
# Heatmap of correlatio matrix
data("diamonds")

# Calculate correlation matrix for numeric variables
cor_matrix <- cor(diamonds[sapply(diamonds, is.numeric)], use = "complete.obs")

#Convert to tidy format
cor_melt <- melt(cor_matrix)

# Create heatmap
ggplot(cor_melt, aes(Var1, Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1, 1), space = "Lab",
                       name = "Correlation") +
  labs(title = "Heatmap of Correlation Matrix for Diamonds Dataset",
       x = "Variables",
       y = "Variables") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 12),
        axis.text.y = element_text(size = 12),
```

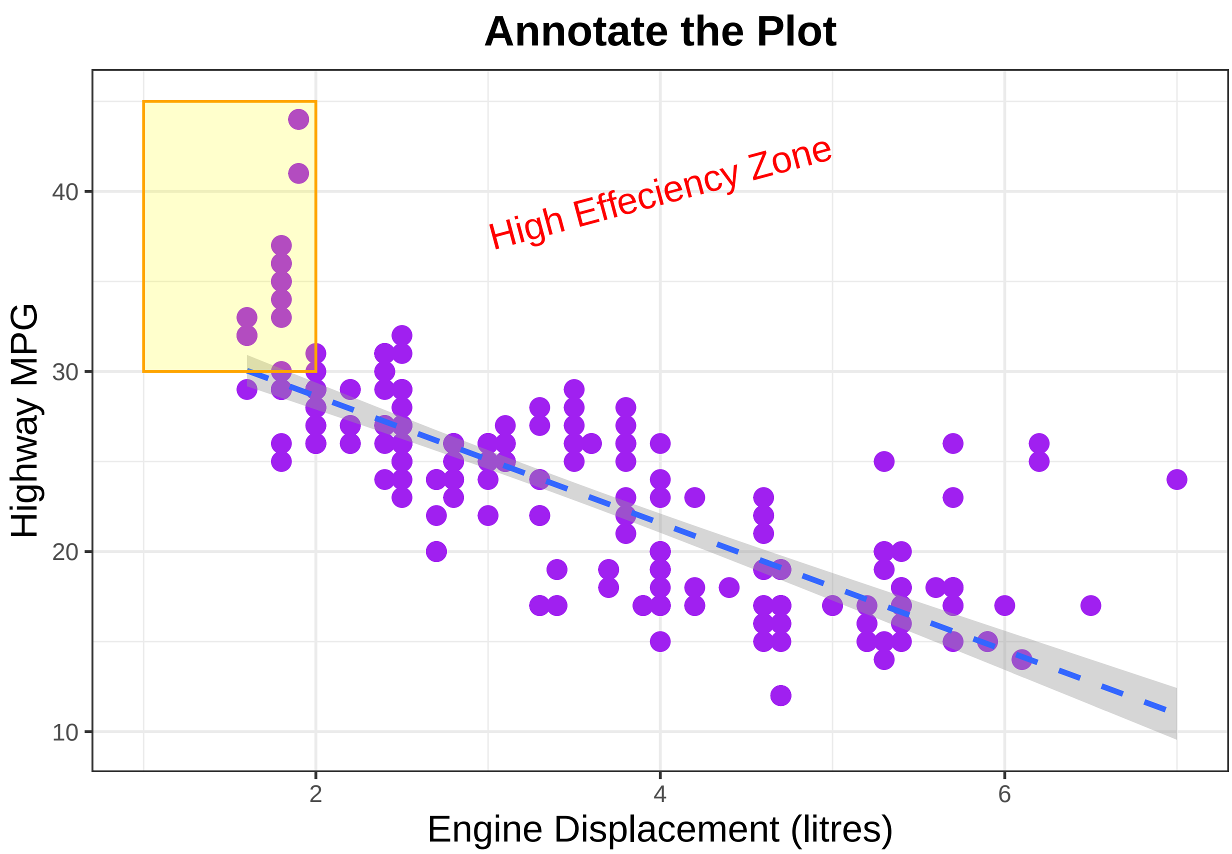
```
        plot.title = element_text(hjust = 0.5, size = 16))
```




```
# Enhancing the scatterplot with annotations
ggplot(mpg, aes(x = displ, y = hwy, fill = class)) +
  geom_point(size = 3, shape = 21, alpha = 0.8) +
  theme_light() +
  scale_color_brewer(palette = "Set2") +
  labs(title = "Customised Scatter Plot",
       x = "Engine Displacement (liters)",
       y = "Highway MPG",
       color = "Vehicle Class") +
  theme(plot.title = element_text(face = "bold", size = 18),
        axis.title = element_text(size = 14),
        legend.background = element_rect(fill = "gray90"))
```




```r
# Annotate plots and save as image files
annotated_plot <- ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point(size = 3, color = "purple") +
  geom_smooth(method = "lm", se = TRUE, linetype = "dashed") +
  labs(title = "Annotate the Plot",
       x = "Engine Displacement (litres)",
       y = "Highway MPG",
       color = "Vehicle Class") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
        axis.title = element_text(size = 14),
        legend.position = "bottom") +
  annotate("text", x = 4, y = 40, label = "High Effeciency Zone",
           color = "red", size = 5, angle = 15) +
  annotate("rect", xmin = 1, xmax = 2, ymin = 30, ymax = 45,
           alpha = 0.2, fill = "yellow", color = "orange")

annotated_plot
```




```
ggsave("annotated_scatterplot.png", plot = annotated_plot, width = 8, height = 6)
```

Program - 7

Linear and Multiple Regression Analysis with Interaction Terms

Date of Execution - 2025-10-14

Objective - This program focuses on regression modeling, interaction effects, and model diagnostics.

```r
# Load necessary libraries
library(MASS)
library(ggplot2)
library(dplyr)
library(caret)
library(car)
library(pROC)
library(corrplot)

# Load the Boston Housing dataset
data("Boston")
head(Boston)
```

```
     crim zn indus chas   nox    rm  age    dis rad tax ptratio
1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3
2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8
3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8
4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7
5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7
6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7
   black lstat medv
1 396.90  4.98 24.0
2 396.90  9.14 21.6
3 392.83  4.03 34.7
4 394.63  2.94 33.4
5 396.90  5.33 36.2
6 394.12  5.21 28.7
```

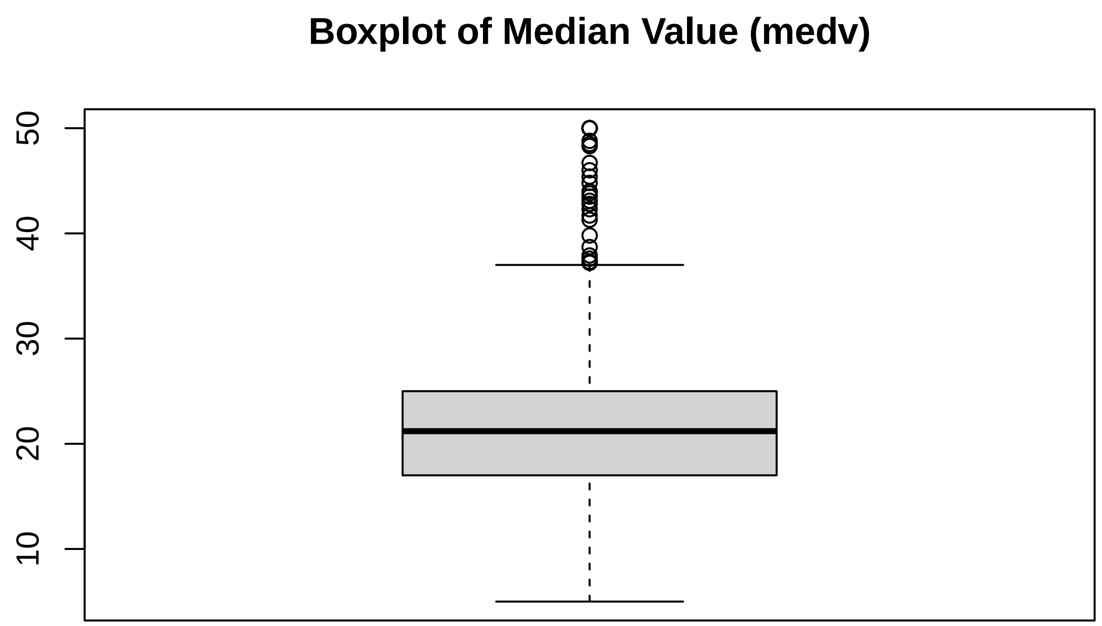
```r
# q1. Preprocessing


# Check for missing values
sum(is.na(Boston))
```

```
[1] 0
```

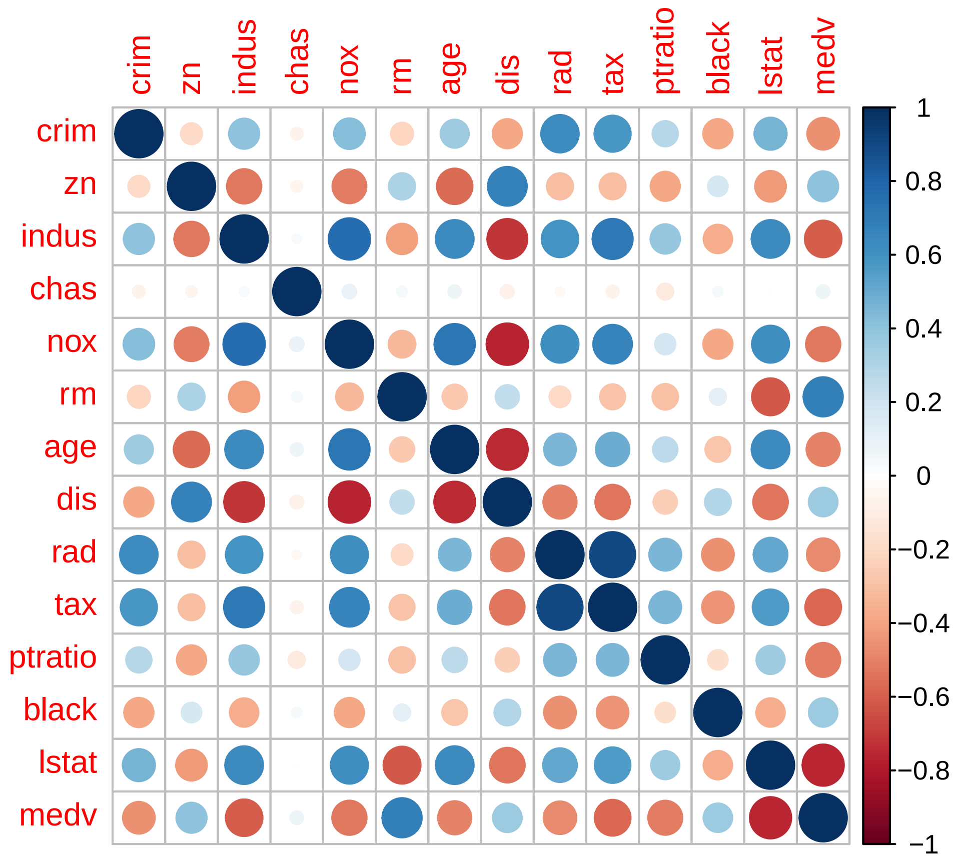
```r
# Summary Statistics
summary(Boston)
```

```
      crim                zn             indus      
 Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46  
 1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19  
 Median : 0.25651   Median :  0.00   Median : 9.69  
 Mean   : 3.61352   Mean   : 11.36   Mean   :11.14  
 3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10  
 Max.   :88.97620   Max.   :100.00   Max.   :27.74  
      chas              nox               rm             age        
 Min.   :0.00000   Min.   :0.3850   Min.   :3.561   Min.   :  2.90  
 1st Qu.:0.00000   1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02  
 Median :0.00000   Median :0.5380   Median :6.208   Median : 77.50  
 Mean   :0.06917   Mean   :0.5547   Mean   :6.285   Mean   : 68.57  
 3rd Qu.:0.00000   3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08  
 Max.   :1.00000   Max.   :0.8710   Max.   :8.780   Max.   :100.00  
      dis              rad              tax           ptratio     
 Min.   : 1.130   Min.   : 1.000   Min.   :187.0   Min.   :12.60  
 1st Qu.: 2.100   1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40  
 Median : 3.207   Median : 5.000   Median :330.0   Median :19.05  
 Mean   : 3.795   Mean   : 9.549   Mean   :408.2   Mean   :18.46  
 3rd Qu.: 5.188   3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20  
 Max.   :12.127   Max.   :24.000   Max.   :711.0   Max.   :22.00  
     black            lstat            medv      
 Min.   :  0.32   Min.   : 1.73   Min.   : 5.00  
 1st Qu.:375.38   1st Qu.: 6.95   1st Qu.:17.02  
 Median :391.44   Median :11.36   Median :21.20  
 Mean   :356.67   Mean   :12.65   Mean   :22.53  
 3rd Qu.:396.23   3rd Qu.:16.95   3rd Qu.:25.00  
 Max.   :396.90   Max.   :37.97   Max.   :50.00  
```

```r
# Check for outliers using boxplots
boxplot(Boston$medv, main = "Boxplot of Median Value (medv)")
```




```r
# Remove potential outliers (optional, based on domain knowledge)
Boston <- Boston %>% filter(medv < 50)

# 2. Feature Selection

# Calculate correlation matrix
corr_matrix <- cor(Boston)
corrplot(corr_matrix, method = "circle")
```




```
# High correlation observed between 'medv', 'lstat', and 'rm'
# We will use 'lstat' and 'rm as predictors based on this analysis.

# 3. Simple Linear Regression Model
simple_model <- lm(medv ~ lstat, data = Boston)
summary(simple_model)
```

```
Call:
lm(formula = medv ~ lstat, data = Boston)

Residuals:
    Min      1Q  Median      3Q     Max
-13.992  -3.313  -0.941   1.914  21.246

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 32.54041    0.48150   67.58   <2e-16 ***
lstat       -0.84374    0.03268  -25.82   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 5.119 on 488 degrees of freedom
Multiple R-squared:  0.5774,    Adjusted R-squared:  0.5765
F-statistic: 666.6 on 1 and 488 DF,  p-value: < 2.2e-16
```

```r
# Interpretation
# - The negative coeffecient for 'lstat' suggests that higher 'lstat' values
#   (higher percentage of lower status population) are associated with lower 'medv'
#   (median home value)
# - The p-value (<0.05) indicates that the relationship is statistically significant.

# 4. Multiple Linear Regression
multiple_model <- lm(medv ~ lstat * rm, data = Boston)
summary(multiple_model)
```

```
Call:
lm(formula = medv ~ lstat * rm, data = Boston)

Residuals:
     Min       1Q   Median       3Q      Max 
-21.3064  -2.4982  -0.3056   1.8635  18.4779 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -25.99970    3.07045  -8.468 2.98e-16 ***
lstat         1.97178    0.17761  11.102  < 2e-16 ***
rm            9.01216    0.46519  19.373  < 2e-16 ***
lstat:rm     -0.43817    0.02976 -14.723  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.845 on 486 degrees of freedom
Multiple R-squared:  0.7625,    Adjusted R-squared:  0.761 
F-statistic:   520 on 3 and 486 DF,  p-value: < 2.2e-16
```

```r
# Interpretation:
# - Significant coeffecients for 'lstat', 'rm', and the interaction term ('lstat:rm')
# - Indicates that the relationship between 'lstat' and 'medv' depends on the value of 'rm'
# - The adjusted R^2 has improved, suggesting better fit when compared to simple model

# 5. Model Performance Evaluation
```

```r
adjusted_R2 <- summary(multiple_model)$adj.r.squared
AIC_Value <- AIC(multiple_model)
BIC_Value <- BIC(multiple_model)

cat("Adjusted R^2: ", adjusted_R2 , "\n")
```

```
Adjusted R^2:  0.7609872
```

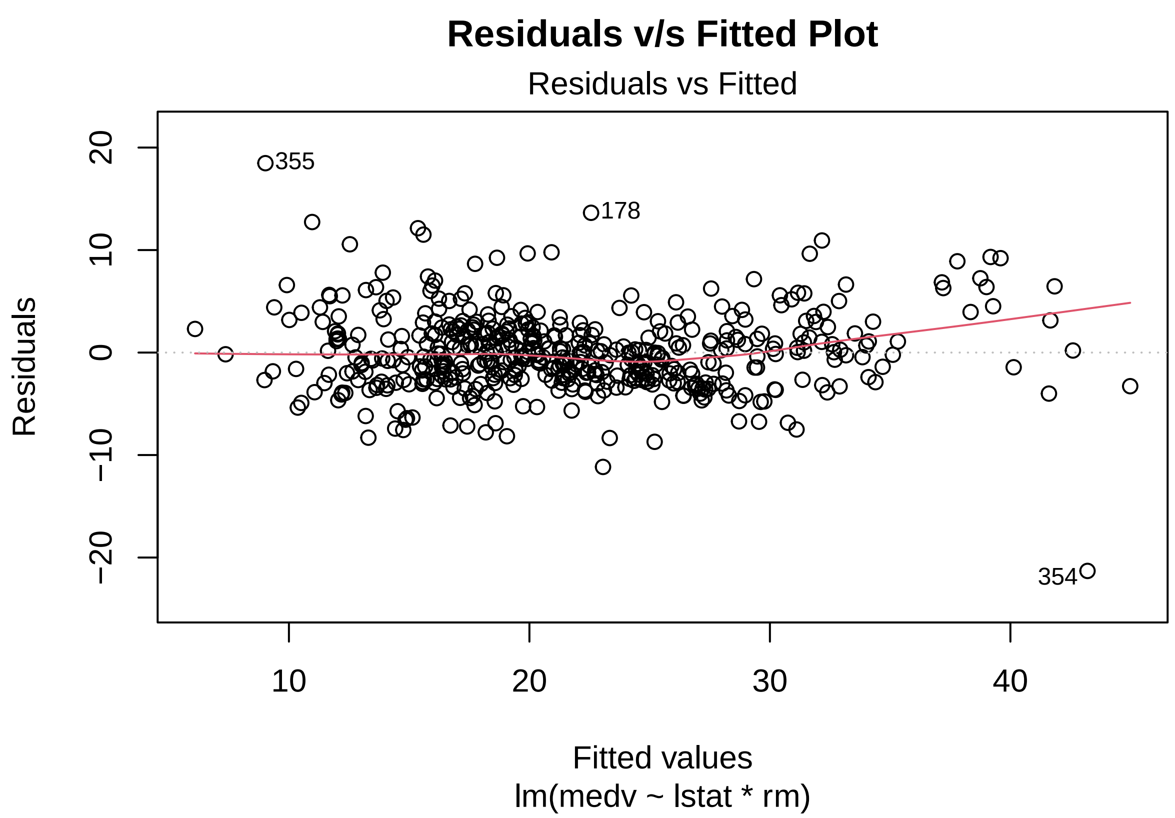
```r
cat("AIC :", AIC_Value , "\n")
```

```
AIC : 2716.448
```

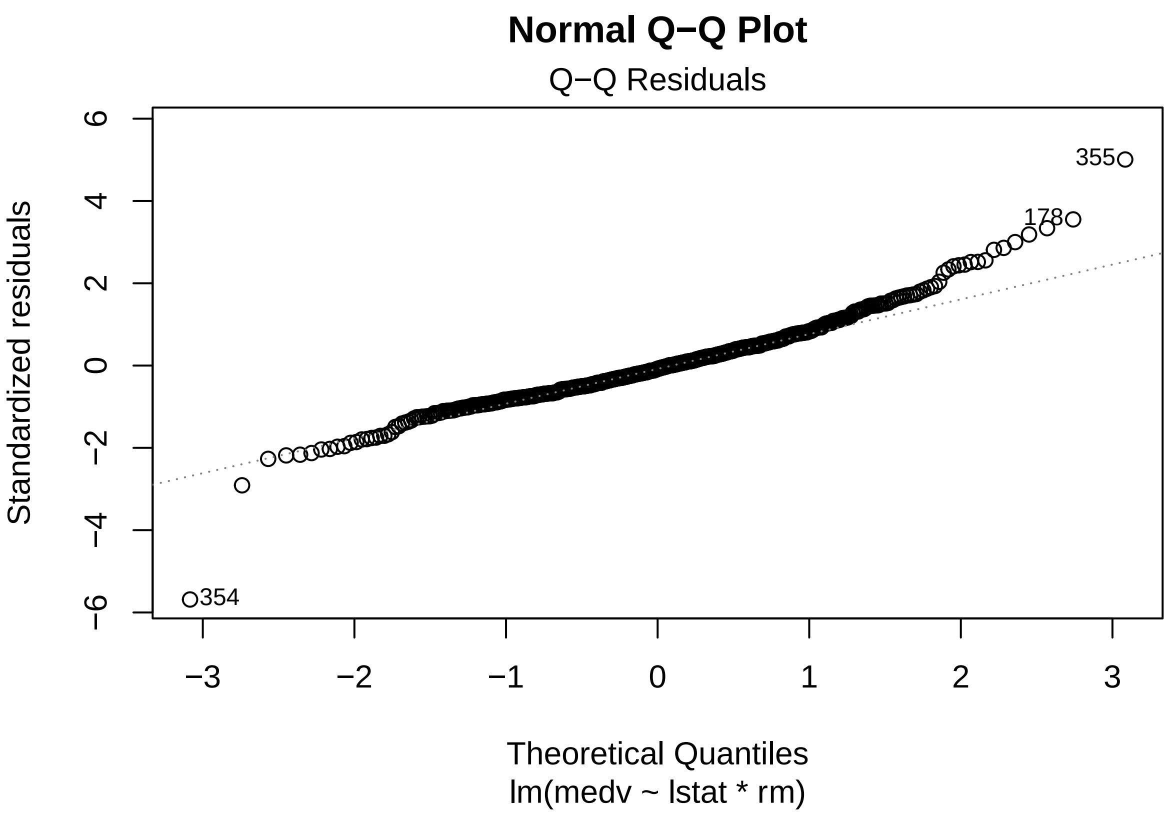
```r
cat("BIC :", BIC_Value , "\n")
```

```
BIC : 2737.42
```

```r
#6. Model Diagnostics : Residual Analysis

# Residual v/s Fitted plot
plot(multiple_model, which = 1, main = "Residuals v/s Fitted Plot")
```



```r
# Q-Q plot for checking normality of residuals
plot(multiple_model, which = 2, main = "Normal Q-Q Plot")
```




```r
# Interpretation
# - The residuals show a random scatter around zero in the Residuals v/s Fitted Plot
# - The Q-Q plot follows a straight line if the residuals are normally distributed

# 7. Cross Validation Model Accuracy
set.seed(123)
train_control <- trainControl(method = "cv", number = 10)
cv_model <- train(medv ~ lstat*rm, data = Boston,
                  method = "lm",
                  trControl = train_control)

# Results
print(cv_model)
```

```
Linear Regression

490 samples
  2 predictor
```

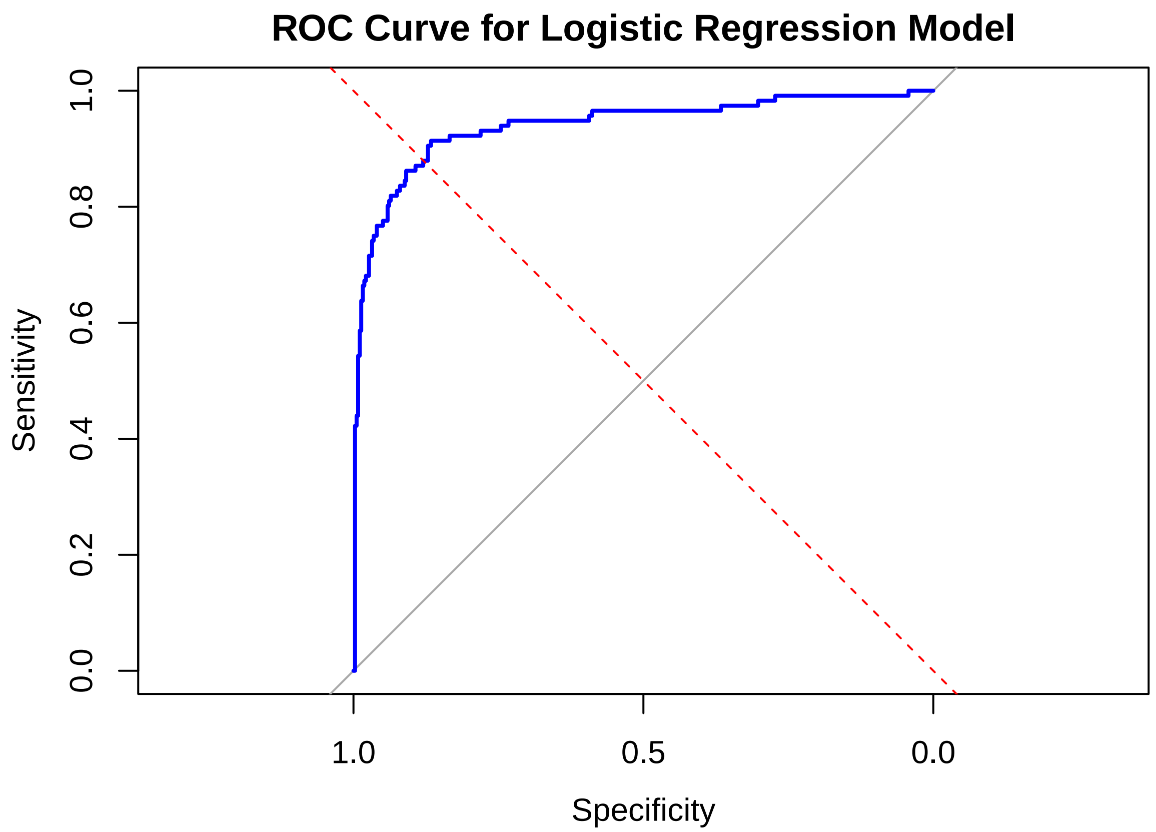
```
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 441, 441, 442, 441, 441, 440, ...
Resampling results:

  RMSE      Rsquared   MAE
  3.855714  0.7597212  2.868657

Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
# Interpretation:
# - Cross Validation RMSE provides a estimate of prediction error
# - Lower RMSE indicates better model performance

# 8. ROC Curve Analysis (Classification Approach)
# Convert 'medv' to a binary classification problem: High (>=25) or Low (<25)
Boston$medv_class <- ifelse(Boston$medv >= 25, 1, 0)

# Fit a logistic regression model for classification
logistic_model <- glm(medv_class ~ lstat*rm, data = Boston, family = "binomial")
summary(logistic_model)
```

```
Call:
glm(formula = medv_class ~ lstat * rm, family = "binomial", data = Boston)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -19.80238    7.45022  -2.658  0.00786 **
lstat        -0.02301    0.75334  -0.031  0.97563
rm            3.29921    1.12727   2.927  0.00343 **
lstat:rm     -0.03989    0.11492  -0.347  0.72851
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 536.34  on 489  degrees of freedom
Residual deviance: 254.70  on 486  degrees of freedom
AIC: 262.7
```

```
Number of Fisher Scoring iterations: 8
```

```r
# Predict probabilities and compute ROC Curve
pred_probs <- predict(logistic_model, type = "response")
roc_curve <- roc(Boston$medv_class, pred_probs)

# Plot ROC Curves
plot(roc_curve, main = "ROC Curve for Logistic Regression Model", col = "blue")
abline(a=0, b=1, lty = 2, col = "red")
```




```r
cat("AUC: ", auc(roc_curve), "\n")
```

```
AUC:  0.9392864
```

```r
# Interpretation
# - The ROC Curve evaluates the trade-off between sensitivity and specificity
# - The Area Under the Curve (AUC) indicates the model's discriminatory ability
#   (AUC closer to 1 is better)
```

Program - 8

K-Means Clustering and PCA for Dimensionality Reduction

Date of Execution - 2025-10-28

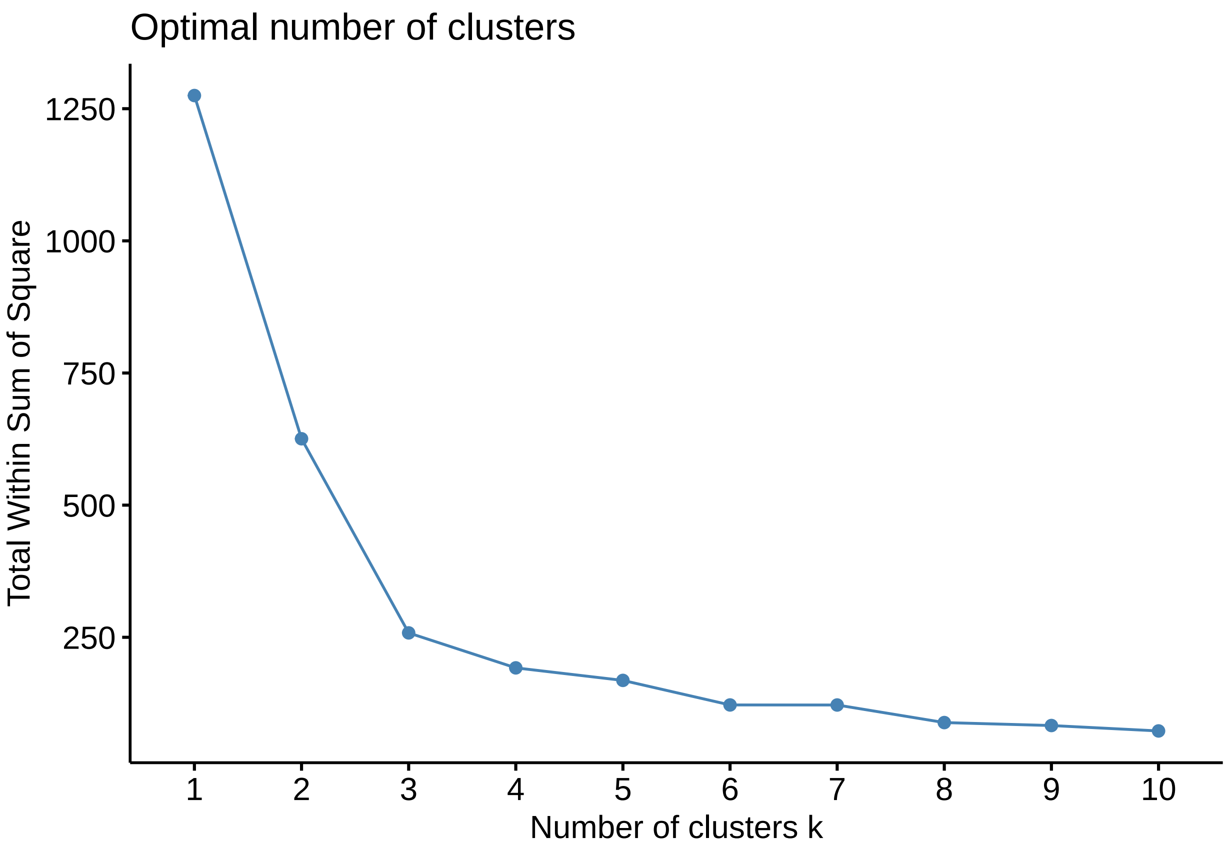
Objective - This program tests the student's knowledge of clustering techniques and dimensionality reduction through PCA.

```
# Load required libraries
library(rattle)      # For Wine dataset
library(ggplot2)     # For visualization
library(cluster)     # For silhouette scores
library(factoextra)  # For PCA and clustering visualization
library(dplyr)       # Often useful for data manipulation

# Normalize function (Min-Max Scaling)
normalize <- function(data) {
  return((data - min(data)) / (max(data) - min(data)))
}

# ---------------------------------------------------------------------
# Analysis for WINE Dataset
# ---------------------------------------------------------------------

# Step 1: Load Wine dataset and normalize
data(wine)
wine_data <- wine[, -1] # Remove the class label
wine_norm <- as.data.frame(lapply(wine_data, normalize))

# Step 2: Apply PCA
wine_pca <- prcomp(wine_norm, scale. = TRUE)
summary(wine_pca)
```

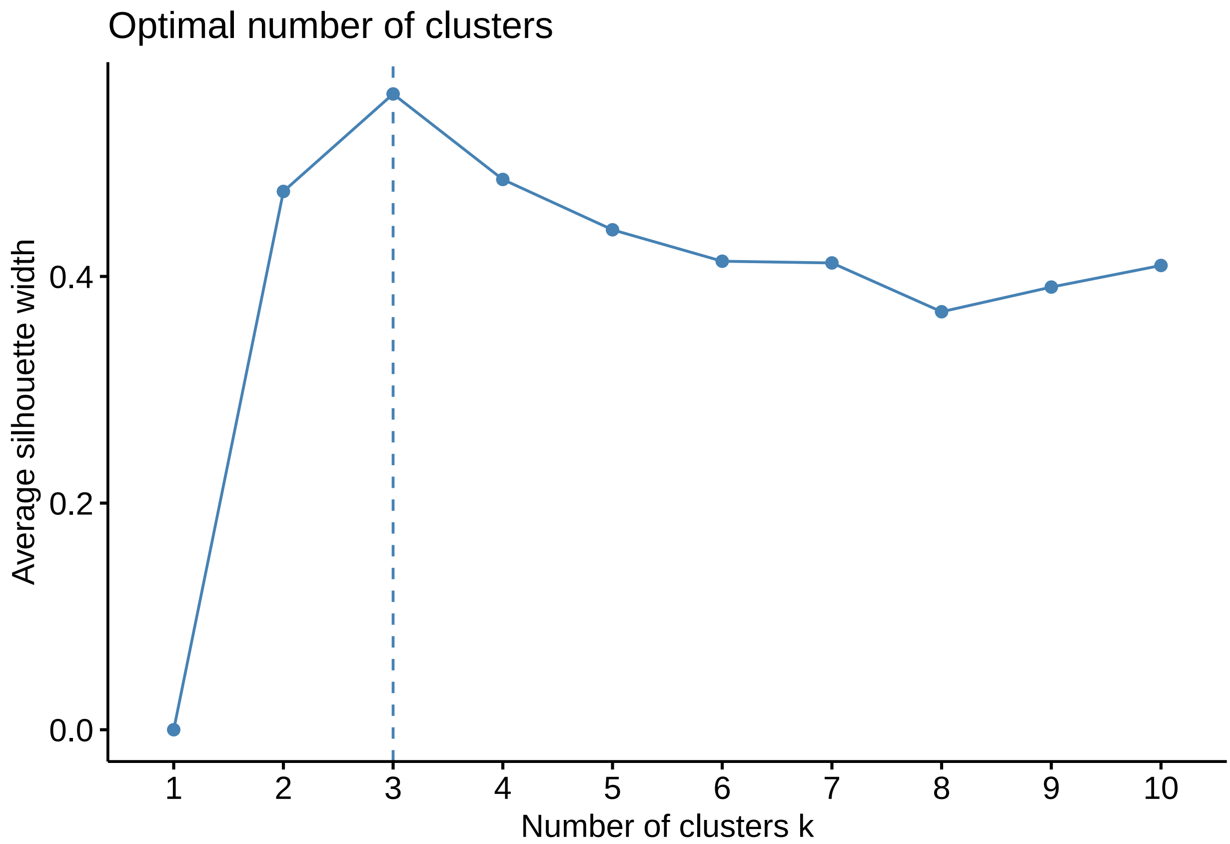
```
Importance of components:
                         PC1    PC2    PC3     PC4     PC5     PC6
Standard deviation     2.169 1.5802 1.2025 0.95863 0.92370 0.80103
Proportion of Variance 0.362 0.1921 0.1112 0.07069 0.06563 0.04936
Cumulative Proportion  0.362 0.5541 0.6653 0.73599 0.80162 0.85098
                           PC7     PC8     PC9    PC10    PC11    PC12
```

```
Standard deviation     0.74231 0.59034 0.53748 0.5009 0.47517 0.41082
Proportion of Variance 0.04239 0.02681 0.02222 0.0193 0.01737 0.01298
Cumulative Proportion  0.89337 0.92018 0.94240 0.9617 0.97907 0.99205
                          PC13
Standard deviation     0.32152
Proportion of Variance 0.00795
Cumulative Proportion  1.00000
```

```r
# Reduce to top 2 principal components
wine_pca_data <- as.data.frame(wine_pca$x[, 1:2])

# Step 3: Determine the optimal number of clusters (Elbow method)
elbow_wine <- fviz_nbclust(wine_pca_data, kmeans, method = "wss")
print(elbow_wine)
```




```r
# Step 4: Silhouette analysis
silhouette_wine <- fviz_nbclust(wine_pca_data, kmeans, method = "silhouette")
print(silhouette_wine)
```




```r
# Step 5: Apply K-means clustering (using centers=3 based on known structure/analysis)
set.seed(123)
wine_kmeans <- kmeans(wine_pca_data, centers = 3, nstart = 25)

# Step 6: Visualize clusters
wine_pca_data$cluster <- as.factor(wine_kmeans$cluster)

p1 <- ggplot(wine_pca_data, aes(x = PC1, y = PC2, color = cluster)) +
  geom_point(size = 3) +
  labs(title = "K-Means Clustering on Wine Dataset")
print(p1)
```


```r
# Step 7: Interpret results
cat("Wine Dataset Clustering Results:\n")
```

```
Wine Dataset Clustering Results:
```

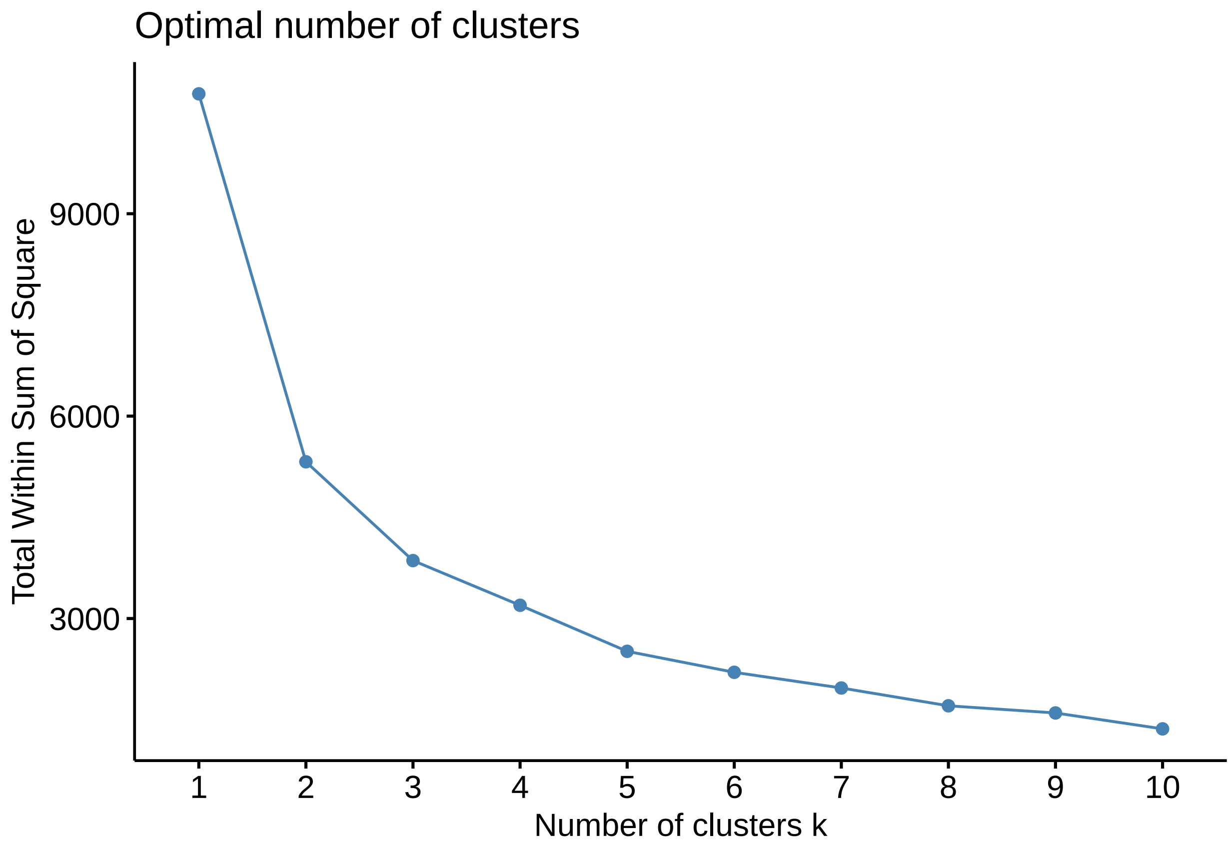
```r
cat("Cluster Sizes: ", wine_kmeans$size, "\n")
```

```
Cluster Sizes:  64 65 49
```

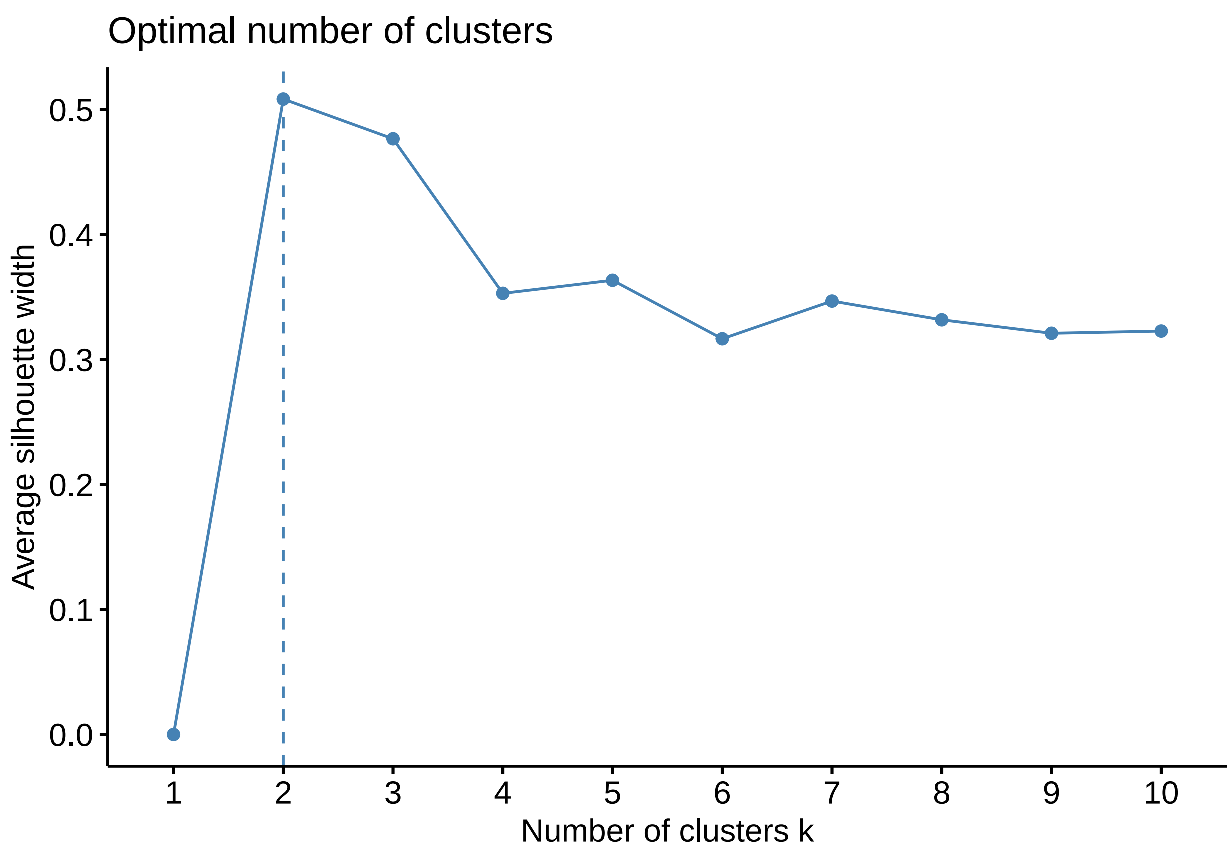
```r
# ----------------------------------------------------------------------
# Step 8: Analysis for Breast Cancer Wisconsin Dataset
# ----------------------------------------------------------------------

# Load dataset from UCI repository (Ensure you have internet connection)
bc_data <- read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/
bc_features <- bc_data[, -c(1, 2)] # Exclude ID and Class columns
bc_norm <- as.data.frame(lapply(bc_features, normalize))

# Apply PCA
bc_pca <- prcomp(bc_norm, scale. = TRUE)
summary(bc_pca)
```

```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6
Standard deviation     3.6444 2.3857 1.67867 1.40735 1.28403 1.09880
Proportion of Variance 0.4427 0.1897 0.09393 0.06602 0.05496 0.04025
Cumulative Proportion  0.4427 0.6324 0.72636 0.79239 0.84734 0.88759
                           PC7     PC8    PC9    PC10   PC11    PC12
Standard deviation     0.82172 0.69037 0.6457 0.59219 0.5421 0.51104
Proportion of Variance 0.02251 0.01589 0.0139 0.01169 0.0098 0.00871
Cumulative Proportion  0.91010 0.92598 0.9399 0.95157 0.9614 0.97007
                          PC13    PC14    PC15    PC16    PC17
Standard deviation     0.49128 0.39624 0.30681 0.28260 0.24372
Proportion of Variance 0.00805 0.00523 0.00314 0.00266 0.00198
Cumulative Proportion  0.97812 0.98335 0.98649 0.98915 0.99113
                          PC18    PC19    PC20   PC21    PC22    PC23
Standard deviation     0.22939 0.22244 0.17652 0.1731 0.16565 0.15602
Proportion of Variance 0.00175 0.00165 0.00104 0.0010 0.00091 0.00081
Cumulative Proportion  0.99288 0.99453 0.99557 0.9966 0.99749 0.99830
                         PC24    PC25    PC26    PC27    PC28    PC29
Standard deviation     0.1344 0.12442 0.09043 0.08307 0.03987 0.02736
Proportion of Variance 0.0006 0.00052 0.00027 0.00023 0.00005 0.00002
Cumulative Proportion  0.9989 0.99942 0.99969 0.99992 0.99997 1.00000
                          PC30
Standard deviation     0.01153
Proportion of Variance 0.00000
Cumulative Proportion  1.00000
```

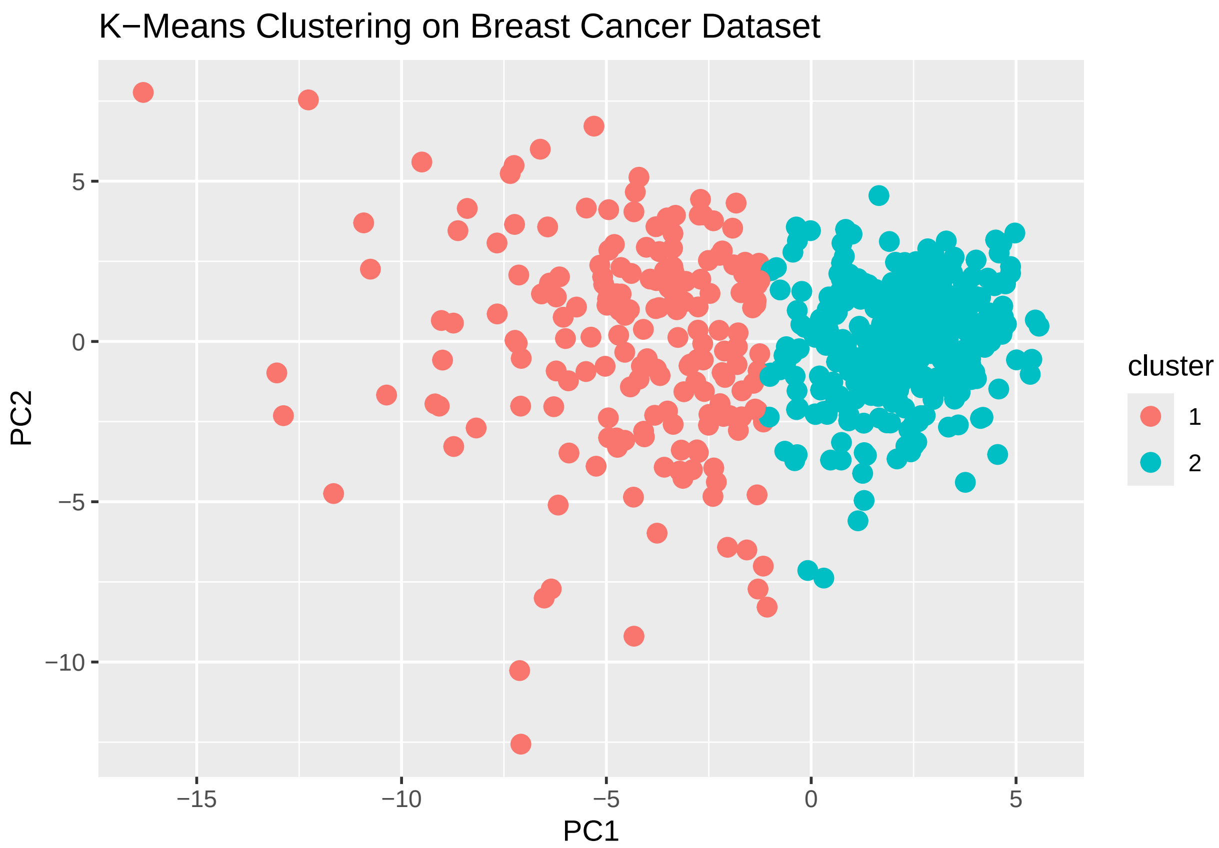
```r
# Reduce to top 2 principal components
bc_pca_data <- as.data.frame(bc_pca$x[, 1:2])

# Optimal number of clusters (Elbow method)
elbow_bc <- fviz_nbclust(bc_pca_data, kmeans, method = "wss")
print(elbow_bc)
```




```r
# Optimal number of clusters (Silhouette analysis)
silhouette_bc <- fviz_nbclust(bc_pca_data, kmeans, method = "silhouette")
print(silhouette_bc)
```



```r
# Apply K-means clustering (using centers=2 based on binary classification/analysis)
set.seed(123)
bc_kmeans <- kmeans(bc_pca_data, centers = 2, nstart = 25)

# Add cluster assignments to PCA data
bc_pca_data$cluster <- as.factor(bc_kmeans$cluster)

# Visualize clusters
p2 <- ggplot(bc_pca_data, aes(x = PC1, y = PC2, color = cluster)) +
  geom_point(size = 3) +
  labs(title = "K-Means Clustering on Breast Cancer Dataset")
print(p2)
```




```r
# Interpret results
cat("Breast Cancer Dataset Clustering Results:\n")
```

```
Breast Cancer Dataset Clustering Results:
```

```r
cat("Cluster Sizes: ", bc_kmeans$size, "\n")
```

```
Cluster Sizes:  191 378
```

Program - 9

Time Series Analysis using ARIMA and Seasonal Decomposition

Date of Execution - 2025-10-28

Objective - This program evaluates students' skills in time series analysis, model fitting, and forecasting.

```r
# Load required libraries for time series analysis and modeling
library(forecast)
library(ggplot2)
library(TSA)
library(tseries)

# Function to perform Exploratory Data Analysis (EDA) on the time series data
perform_eda <- function(ts_data, dataset_name) {
  cat(" Exploratory Data Analysis for ", dataset_name, "\n")
  print(summary(ts_data)) # Print summary of the dataset
  plot(ts_data, main = paste(dataset_name, " Time Series "), ylab = " Values ", xlab = "Time ")
  cat("ACF and PACF plots :\n")
  acf(ts_data, main = paste("ACF of", dataset_name)) # Autocorrelation plot
  pacf(ts_data, main = paste(" PACF of", dataset_name)) # Partial autocorrelation plot
}

# Function to decompose the time series into trend , seasonal , and residual components
decompose_ts <- function(ts_data, dataset_name) {
  cat(" Decomposing the time series for ", dataset_name, "\n")
  decomposition <- decompose(ts_data) # Decompose the time series
  plot(decomposition) # Plot the decomposition
  return(decomposition) # Return the decomposition result
}

# Function to fit an ARIMA model to the time series data
fit_arima <- function(ts_data, dataset_name) {
  cat(" Fitting ARIMA model for ", dataset_name, "\n")
  adf_test <- adf.test(ts_data, alternative = "stationary") # ADF test for stationarity
  cat("ADF Test p- value :", adf_test$p.value, "\n")
  # If p- value > 0.05 , data is non - stationary , so we difference the data
  if (adf_test$p.value > 0.05) {
    ts_data <- diff(ts_data) # Difference the data to make it stationary
```

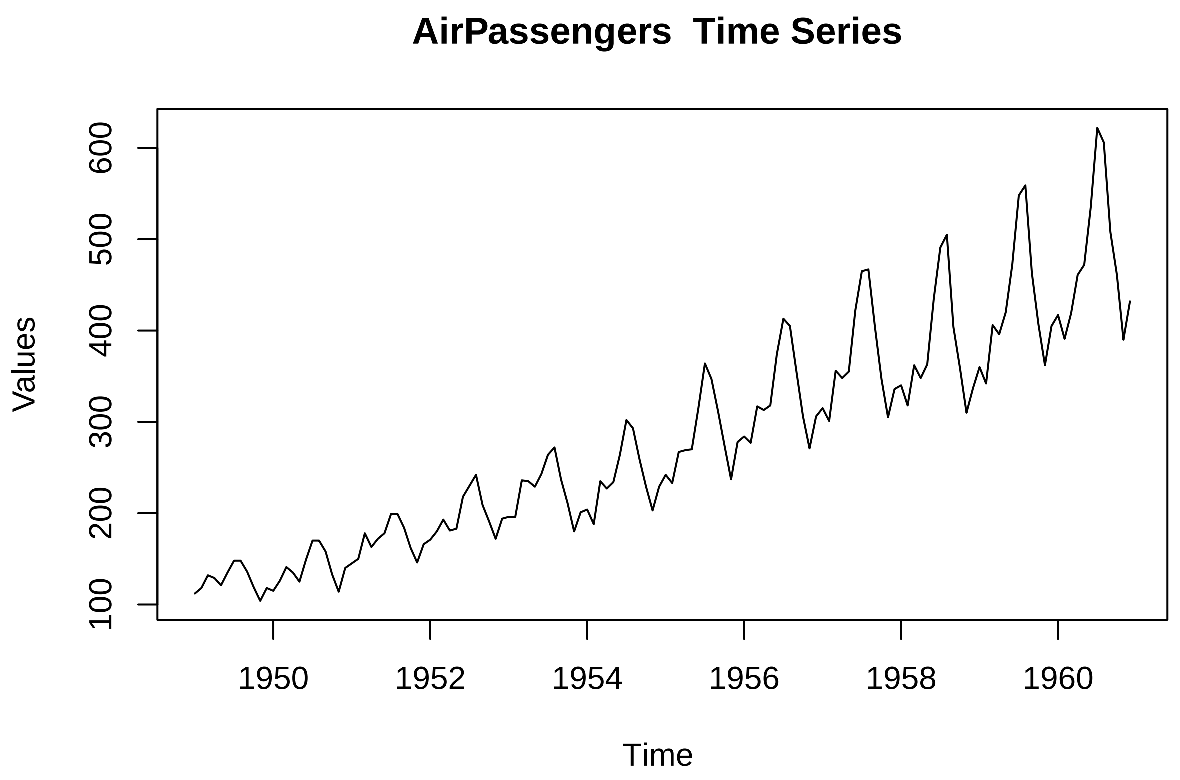
```r
    plot(ts_data, main = paste(dataset_name, " Differenced Time Series "))
  }
  auto_model <- auto.arima(ts_data, seasonal = FALSE) # Fit ARIMA model (non - seasonal)
  print(summary(auto_model)) # Print ARIMA model summary
  forecast_result <- forecast(auto_model, h = 12) # Forecast next 12 periods
  plot(forecast_result, main = paste(dataset_name, " ARIMA Forecast ")) # Plot ARIMA forecast
  return(auto_model) # Return the fitted ARIMA model
}

# Function to fit a Seasonal ARIMA ( SARIMA ) model to the time series data
fit_sarima <- function(ts_data, dataset_name) {
  cat(" Fitting SARIMA model for ", dataset_name, "\n")
  auto_sarima <- auto.arima(ts_data, seasonal = TRUE) # Fit SARIMA model ( seasonal )
  print(summary(auto_sarima)) # Print SARIMA model summary
  sarima_forecast <- forecast(auto_sarima, h = 12) # Forecast next 12 periods
  plot(sarima_forecast, main = paste(dataset_name, " SARIMA Forecast ")) # Plot SARIMA forecast
  return(auto_sarima) # Return the fitted SARIMA model
}

# Function to compare ARIMA and SARIMA models by evaluating forecast accuracy
compare_models <- function(arima_model, sarima_model, ts_data) {
  cat(" Comparing ARIMA and SARIMA models :\n")
  h <- min(12, length(ts_data)) # Forecast horizon of 12 or adjusted based on dataset length
  arima_forecast <- forecast(arima_model, h = h) # ARIMA forecast
  sarima_forecast <- forecast(sarima_model, h = h) # SARIMA forecast
  actual_values <- ts_data[(length(ts_data) - h + 1): length(ts_data)] # Comparison

  # Calculate accuracy of both models
  arima_accuracy <- accuracy(arima_forecast$mean, actual_values)
  sarima_accuracy <- accuracy(sarima_forecast$mean, actual_values)
  cat(" ARIMA Forecast Accuracy :\n", arima_accuracy) # Print ARIMA accuracy
  cat(" SARIMA Forecast Accuracy :\n", sarima_accuracy) # Print SARIMA accuracy
}

# Function to visualize the comparison of ARIMA and SARIMA forecast performance
plot_forecast_comparison <- function(actual_values, arima_forecast, sarima_forecast, time_points) {
  arima_rmse <- sqrt(mean((arima_forecast - actual_values)^2)) # Calculate RMSE for ARIMA
  sarima_rmse <- sqrt(mean((sarima_forecast - actual_values)^2)) # Calculate RMSE for SARIMA

  # Color coding for better and worse RMSE
```

```r
  better_color <- ifelse(arima_rmse < sarima_rmse, "green", "red")
  worse_color <- ifelse(arima_rmse < sarima_rmse, "red", "green")

  # Plot actual values and forecasts
  plot(time_points, actual_values, type = "o", col = "blue", pch = 16, lty = 1, xlab = " Time ",
       ylab = " Values ", main = " Forecast Comparison ")
  lines(time_points, arima_forecast, col = better_color, lty = 2, lwd = 2) # ARIMA forecast line
  lines(time_points, sarima_forecast, col = worse_color, lty = 3, lwd = 2) # SARIMA forecast line

  # Add a legend to the plot
  legend("topright", legend = c("Actual Values", paste("ARIMA (RMSE =", round(arima_rmse, 2), ")"),
                                paste("SARIMA (RMSE =", round(sarima_rmse, 2), ")")),
         col = c("blue", better_color, worse_color), lty = c(1, 2, 3), lwd = c(1, 2, 2),
         pch = c(16, NA, NA))
}

# AirPassengers Dataset Analysis
data("AirPassengers")
air_data <- AirPassengers
cat("\n- - - AirPassengers Dataset - - -\n")
```

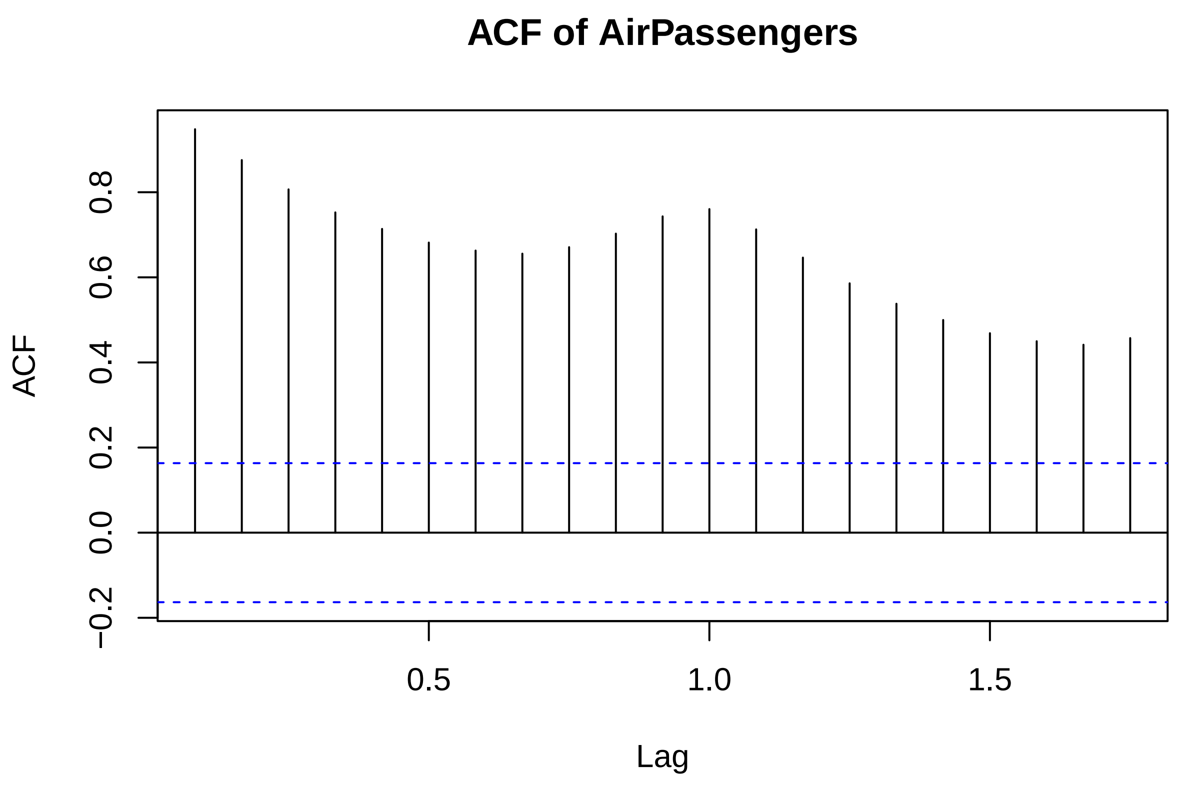
```
- - - AirPassengers Dataset - - -
```

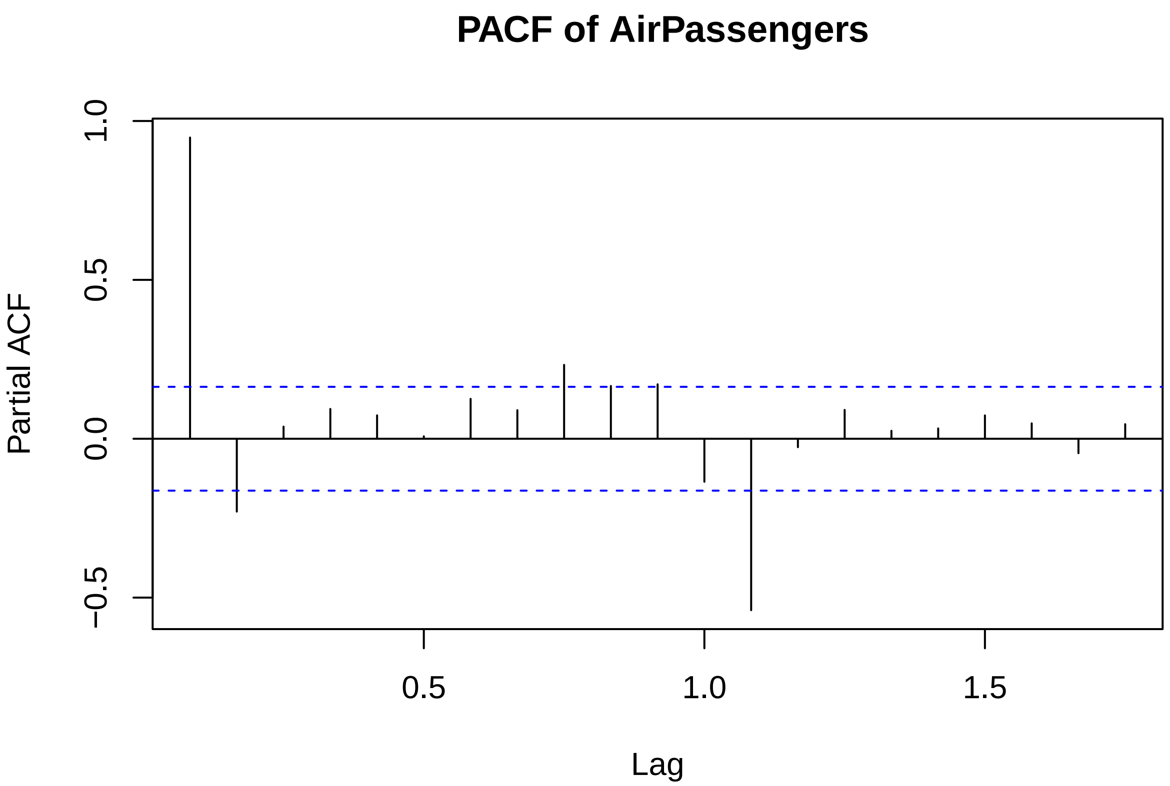
```r
perform_eda(air_data, "AirPassengers")
```

```
 Exploratory Data Analysis for  AirPassengers
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  104.0   180.0   265.5   280.3   360.5   622.0
```



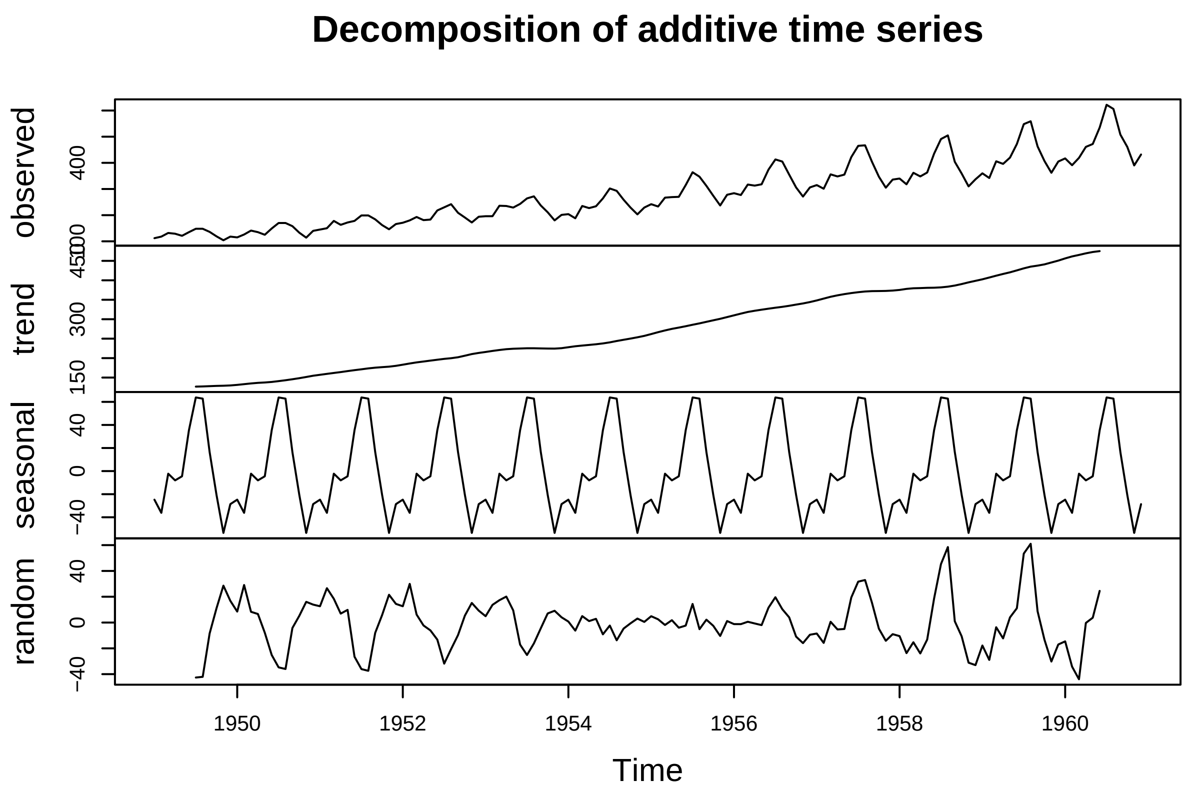
ACF and PACF plots :






```r
decompose_ts(air_data, "AirPassengers")
```

```
Decomposing the time series for  AirPassengers
```



```
$x
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432

$seasonal
            Jan        Feb        Mar        Apr        May
1949 -24.748737 -36.188131  -2.241162  -8.036616  -4.506313
1950 -24.748737 -36.188131  -2.241162  -8.036616  -4.506313
1951 -24.748737 -36.188131  -2.241162  -8.036616  -4.506313
1952 -24.748737 -36.188131  -2.241162  -8.036616  -4.506313
1953 -24.748737 -36.188131  -2.241162  -8.036616  -4.506313
1954 -24.748737 -36.188131  -2.241162  -8.036616  -4.506313
1955 -24.748737 -36.188131  -2.241162  -8.036616  -4.506313
1956 -24.748737 -36.188131  -2.241162  -8.036616  -4.506313
1957 -24.748737 -36.188131  -2.241162  -8.036616  -4.506313
1958 -24.748737 -36.188131  -2.241162  -8.036616  -4.506313
1959 -24.748737 -36.188131  -2.241162  -8.036616  -4.506313
1960 -24.748737 -36.188131  -2.241162  -8.036616  -4.506313
            Jun        Jul        Aug        Sep        Oct
1949  35.402778  63.830808  62.823232  16.520202 -20.642677
1950  35.402778  63.830808  62.823232  16.520202 -20.642677
1951  35.402778  63.830808  62.823232  16.520202 -20.642677
1952  35.402778  63.830808  62.823232  16.520202 -20.642677
1953  35.402778  63.830808  62.823232  16.520202 -20.642677
1954  35.402778  63.830808  62.823232  16.520202 -20.642677
1955  35.402778  63.830808  62.823232  16.520202 -20.642677
1956  35.402778  63.830808  62.823232  16.520202 -20.642677
1957  35.402778  63.830808  62.823232  16.520202 -20.642677
1958  35.402778  63.830808  62.823232  16.520202 -20.642677
1959  35.402778  63.830808  62.823232  16.520202 -20.642677
```

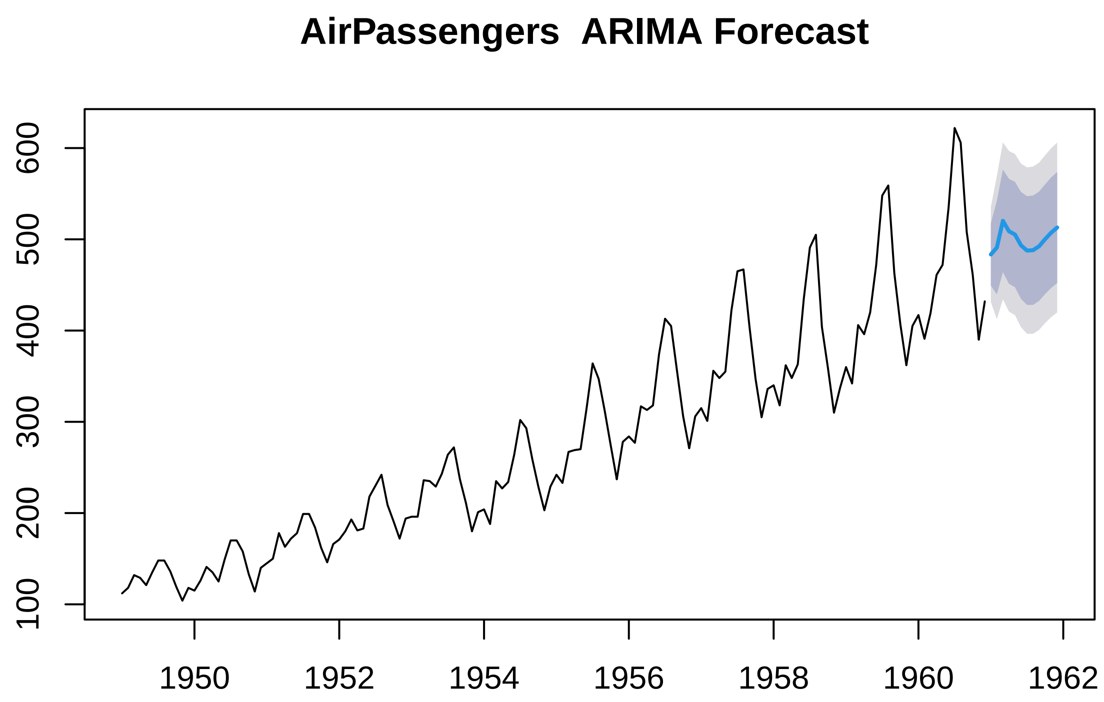
```
1960  35.402778  63.830808  62.823232  16.520202 -20.642677
            Nov        Dec
1949 -53.593434 -28.619949
1950 -53.593434 -28.619949
1951 -53.593434 -28.619949
1952 -53.593434 -28.619949
1953 -53.593434 -28.619949
1954 -53.593434 -28.619949
1955 -53.593434 -28.619949
1956 -53.593434 -28.619949
1957 -53.593434 -28.619949
1958 -53.593434 -28.619949
1959 -53.593434 -28.619949
1960 -53.593434 -28.619949

$trend
          Jan      Feb      Mar      Apr      May      Jun      Jul
1949       NA       NA       NA       NA       NA       NA 126.7917
1950 131.2500 133.0833 134.9167 136.4167 137.4167 138.7500 140.9167
1951 157.1250 159.5417 161.8333 164.1250 166.6667 169.0833 171.2500
1952 183.1250 186.2083 189.0417 191.2917 193.5833 195.8333 198.0417
1953 215.8333 218.5000 220.9167 222.9167 224.0833 224.7083 225.3333
1954 228.0000 230.4583 232.2500 233.9167 235.6250 237.7500 240.5000
1955 261.8333 266.6667 271.1250 275.2083 278.5000 281.9583 285.7500
1956 309.9583 314.4167 318.6250 321.7500 324.5000 327.0833 329.5417
1957 348.2500 353.0000 357.6250 361.3750 364.5000 367.1667 369.4583
1958 375.2500 377.9167 379.5000 380.0000 380.7083 380.9583 381.8333
1959 402.5417 407.1667 411.8750 416.3333 420.5000 425.5000 430.7083
1960 456.3333 461.3750 465.2083 469.3333 472.7500 475.0417       NA
          Aug      Sep      Oct      Nov      Dec
1949 127.2500 127.9583 128.5833 129.0000 129.7500
1950 143.1667 145.7083 148.4167 151.5417 154.7083
1951 173.5833 175.4583 176.8333 178.0417 180.1667
1952 199.7500 202.2083 206.2500 210.4167 213.3750
1953 225.3333 224.9583 224.5833 224.4583 225.5417
1954 243.9583 247.1667 250.2500 253.5000 257.1250
1955 289.3333 293.2500 297.1667 301.0000 305.4583
1956 331.8333 334.4583 337.5417 340.5417 344.0833
1957 371.2083 372.1667 372.4167 372.7500 373.6250
1958 383.6667 386.5000 390.3333 394.7083 398.6250
1959 435.1250 437.7083 440.9583 445.8333 450.6250
```

```
1960        NA       NA       NA       NA       NA

$random
             Jan         Feb         Mar         Apr         May
1949          NA          NA          NA          NA          NA
1950   8.4987374  29.1047980   8.3244949   6.6199495  -7.9103535
1951  12.6237374  26.6464646  18.4078283   6.9116162   9.8396465
1952  12.6237374  29.9797980   6.1994949  -2.2550505  -6.0770202
1953   4.9154040  13.6881313  17.3244949  20.1199495   9.4229798
1954   0.7487374  -6.2702020   4.9911616   1.1199495   2.8813131
1955   4.9154040   2.5214646  -1.8838384   1.8282828  -3.9936869
1956  -1.2095960  -1.2285354   0.6161616  -0.7133838  -1.9936869
1957  -8.5012626 -15.8118687   0.6161616  -5.3383838  -4.9936869
1958 -10.5012626 -23.7285354 -15.2588384 -23.9633838 -13.2020202
1959 -17.7929293 -28.9785354  -3.6338384 -12.2967172   4.0063131
1960 -14.5845960 -34.1868687 -43.9671717  -0.2967172   3.7563131
             Jun         Jul         Aug         Sep         Oct
1949          NA -42.6224747 -42.0732323  -8.4785354  11.0593434
1950 -25.1527778 -34.7474747 -35.9898990  -4.2285354   5.2260101
1951 -26.4861111 -36.0808081 -37.4065657  -7.9785354   5.8093434
1952 -13.2361111 -31.8724747 -20.5732323  -9.7285354   5.3926768
1953 -17.1111111 -25.1641414 -16.1565657  -4.4785354   7.0593434
1954  -9.1527778  -2.3308081 -13.7815657  -4.6868687  -0.6073232
1955  -2.3611111  14.4191919  -5.1565657   2.2297980  -2.5239899
1956  11.5138889  19.6275253  10.3434343   4.0214646 -10.8989899
1957  19.4305556  31.7108586  32.9684343  15.3131313  -4.7739899
1958  18.6388889  45.3358586  58.5101010   0.9797980 -10.6906566
1959  11.0972222  53.4608586  61.0517677   8.7714646 -13.3156566
1960  24.5555556          NA          NA          NA          NA
             Nov         Dec
1949  28.5934343  16.8699495
1950  16.0517677  13.9116162
1951  21.5517677  14.4532828
1952  15.1767677   9.2449495
1953   9.1351010   4.0782828
1954   3.0934343   0.4949495
1955 -10.4065657   1.1616162
1956 -15.9482323  -9.4633838
1957 -14.1565657  -9.0050505
1958 -31.1148990 -33.0050505
1959 -30.2398990 -17.0050505
```

```
1960          NA          NA

$figure
 [1] -24.748737 -36.188131  -2.241162  -8.036616  -4.506313  35.402778
 [7]  63.830808  62.823232  16.520202 -20.642677 -53.593434 -28.619949

$type
[1] "additive"

attr(,"class")
[1] "decomposed.ts"
```

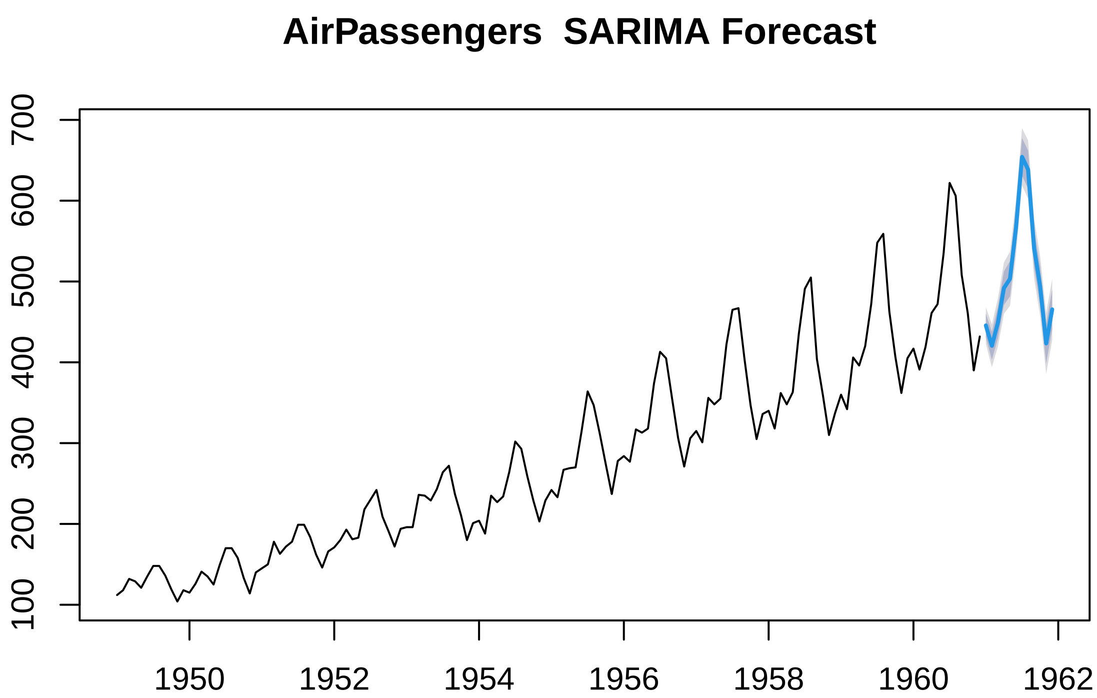
```r
arima_air <- fit_arima(air_data, "AirPassengers")
```

```
 Fitting ARIMA model for  AirPassengers

Warning in adf.test(ts_data, alternative = "stationary"): p-value
smaller than printed p-value

ADF Test p- value : 0.01
Series: ts_data
ARIMA(4,1,2) with drift

Coefficients:
         ar1     ar2      ar3      ar4      ma1      ma2   drift
      0.2243  0.3689  -0.2567  -0.2391  -0.0971  -0.8519  2.6809
s.e.  0.1047  0.1147   0.0985   0.0919   0.0866   0.0877  0.1711

sigma^2 = 706.3:  log likelihood = -670.07
AIC=1356.15   AICc=1357.22   BIC=1379.85

Training set error measures:
                    ME     RMSE      MAE       MPE     MAPE      MASE
Training set -1.228696 25.82793 20.59211 -1.665245 7.476447 0.6428946
                     ACF1
Training set 0.0009861078
```




```r
sarima_air <- fit_sarima(air_data, "AirPassengers")
```

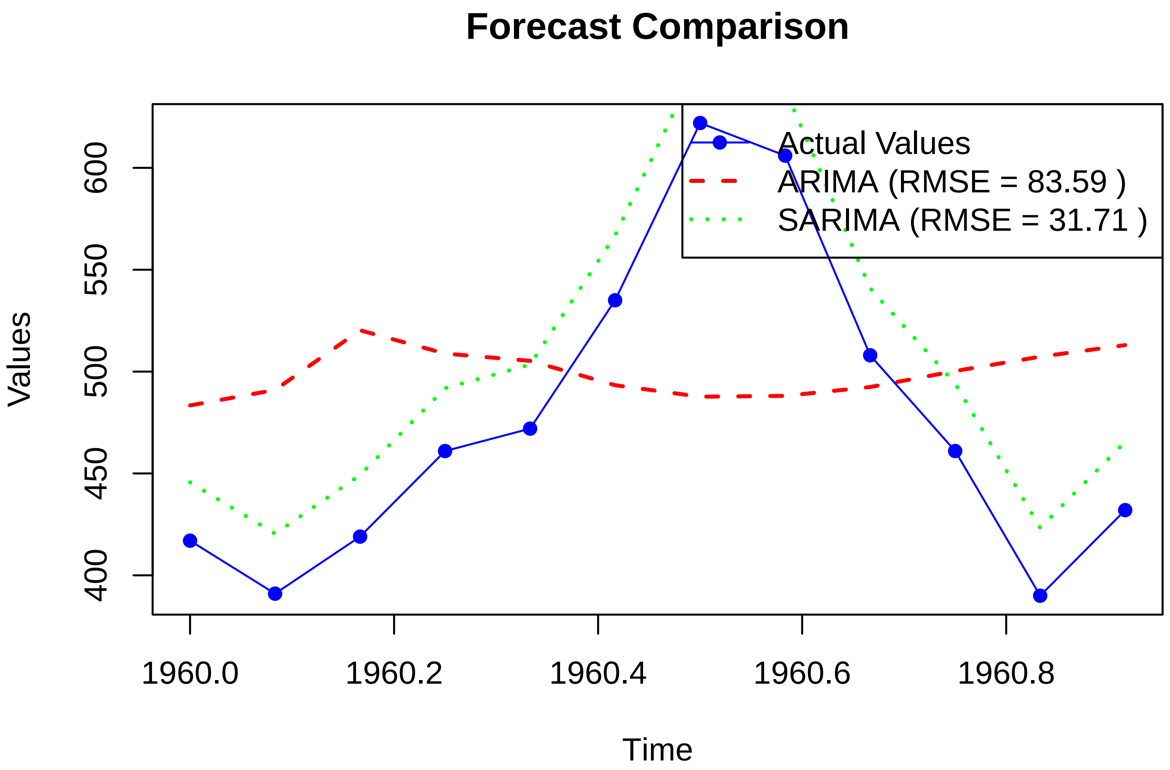
```
 Fitting SARIMA model for  AirPassengers
Series: ts_data
ARIMA(2,1,1)(0,1,0)[12]

Coefficients:
         ar1     ar2      ma1
      0.5960  0.2143  -0.9819
s.e.  0.0888  0.0880   0.0292

sigma^2 = 132.3:  log likelihood = -504.92
AIC=1017.85   AICc=1018.17   BIC=1029.35

Training set error measures:
                 ME     RMSE     MAE      MPE     MAPE     MASE
Training set 1.3423 10.84619 7.86754 0.420698 2.800458 0.245628
                   ACF1
Training set -0.00124847
```




```
compare_models(arima_air, sarima_air, air_data)
```

```
Comparing ARIMA and SARIMA models :
ARIMA Forecast Accuracy :
-23.06901 83.58979 74.65149 -7.376947 16.02863 SARIMA Forecast Accuracy :
-31.66945 31.70666 31.66945 -6.784721 6.784721
```

```
# Forecasting and plot comparison for AirPassengers dataset
h_air <- 12 # Define forecast horizon for AirPassengers dataset (12 months ahead)
# Extract the actual values for the last 12 months of the AirPassengers data
air_actual_values <- air_data[(length(air_data) - h_air + 1): length(air_data)]
# Generate ARIMA forecast for the next 12 months
arima_air_forecast <- forecast(arima_air, h = h_air)$mean
# Generate SARIMA forecast for the next 12 months
sarima_air_forecast <- forecast(sarima_air, h = h_air)$mean
# Extract the time points for the last 12 months
time_points_air <- time(air_data)[(length(air_data) - h_air + 1): length(air_data)]
# Plot and compare the forecasts from ARIMA and SARIMA models against the actual values
plot_forecast_comparison(air_actual_values, arima_air_forecast, sarima_air_forecast,
                         time_points_air)
```

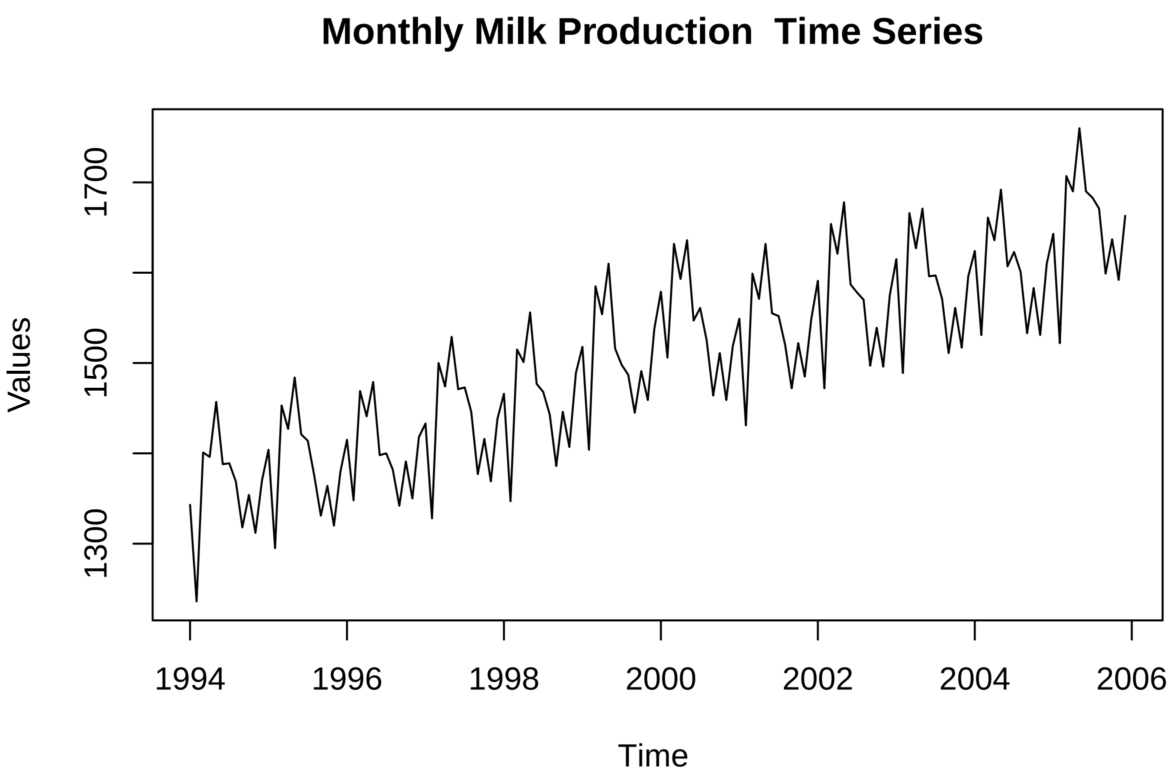



```r
# Monthly Milk Production Dataset Analysis
data(milk) # Load the Monthly Milk Production dataset
milk_data <- milk # Assign the dataset to a variable
cat("\n- - - Monthly Milk Production Dataset - - -\n")
```

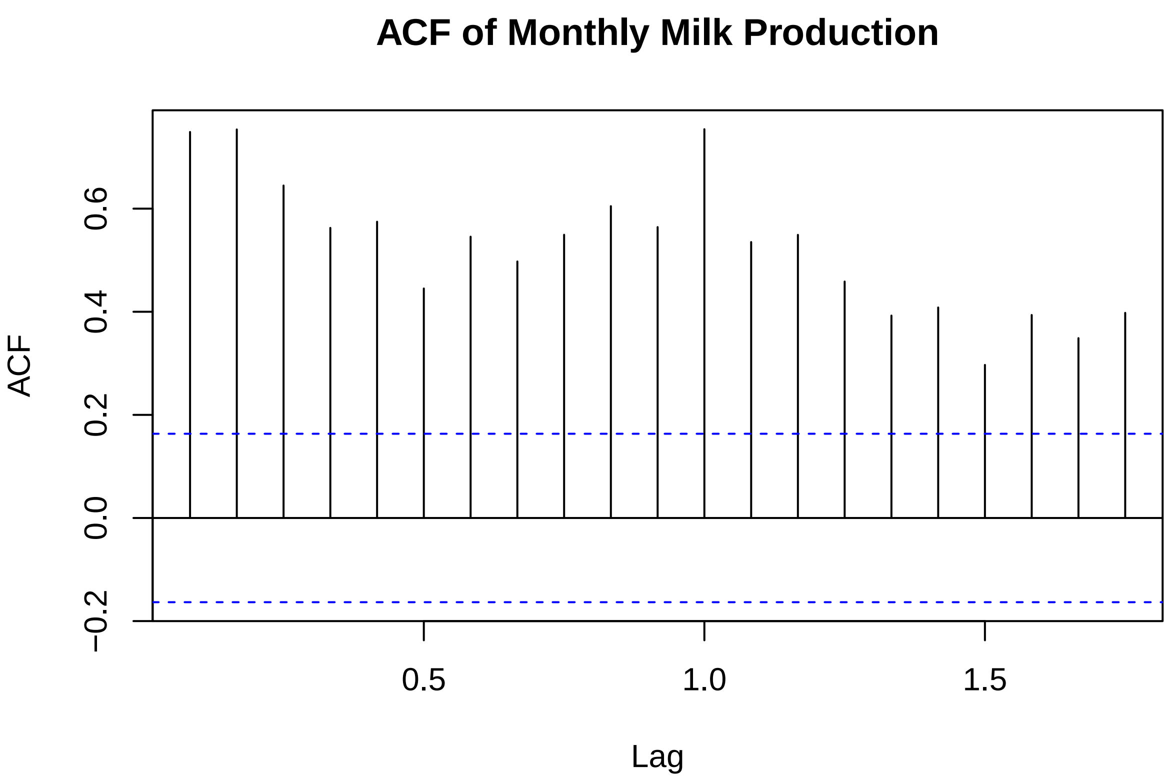
```
- - - Monthly Milk Production Dataset - - -
```

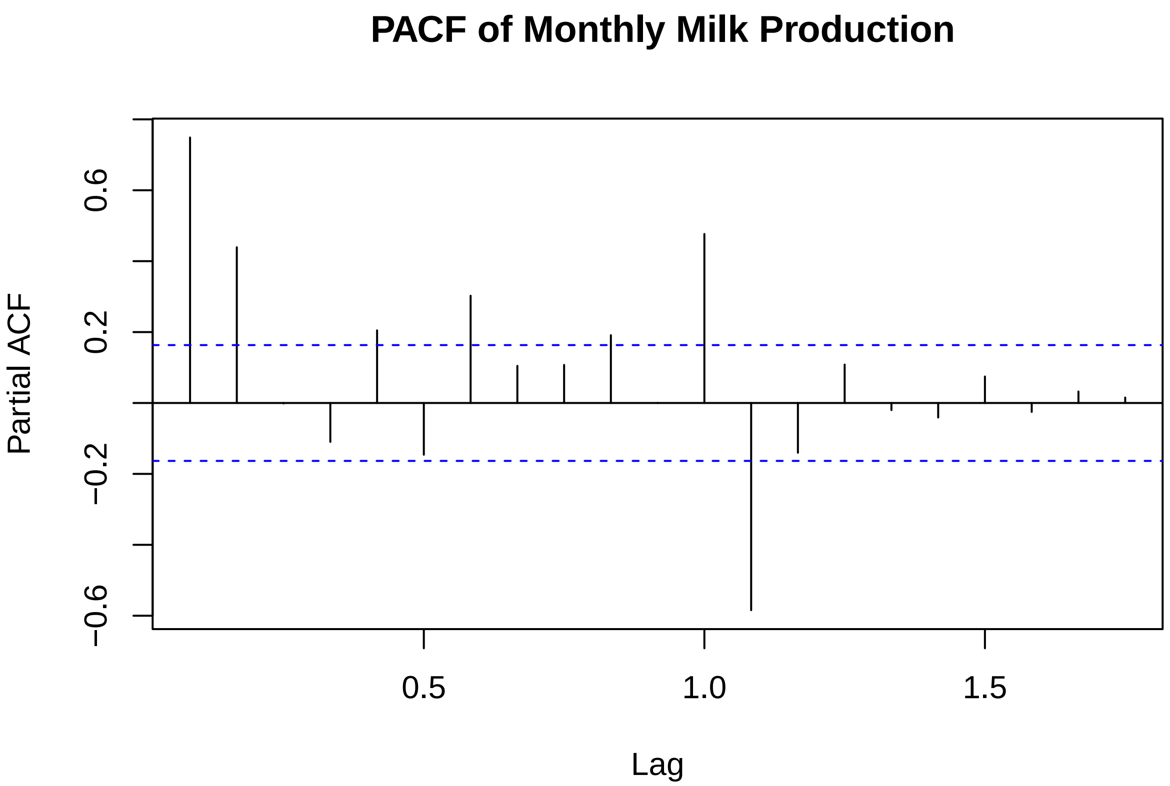
```r
# Perform Exploratory Data Analysis (EDA) for the Milk Production dataset
perform_eda(milk_data, "Monthly Milk Production")
```

```
 Exploratory Data Analysis for  Monthly Milk Production
      milk
 Min.   :1236
 1st Qu.:1420
 Median :1504
 Mean   :1504
 3rd Qu.:1588
 Max.   :1760
```



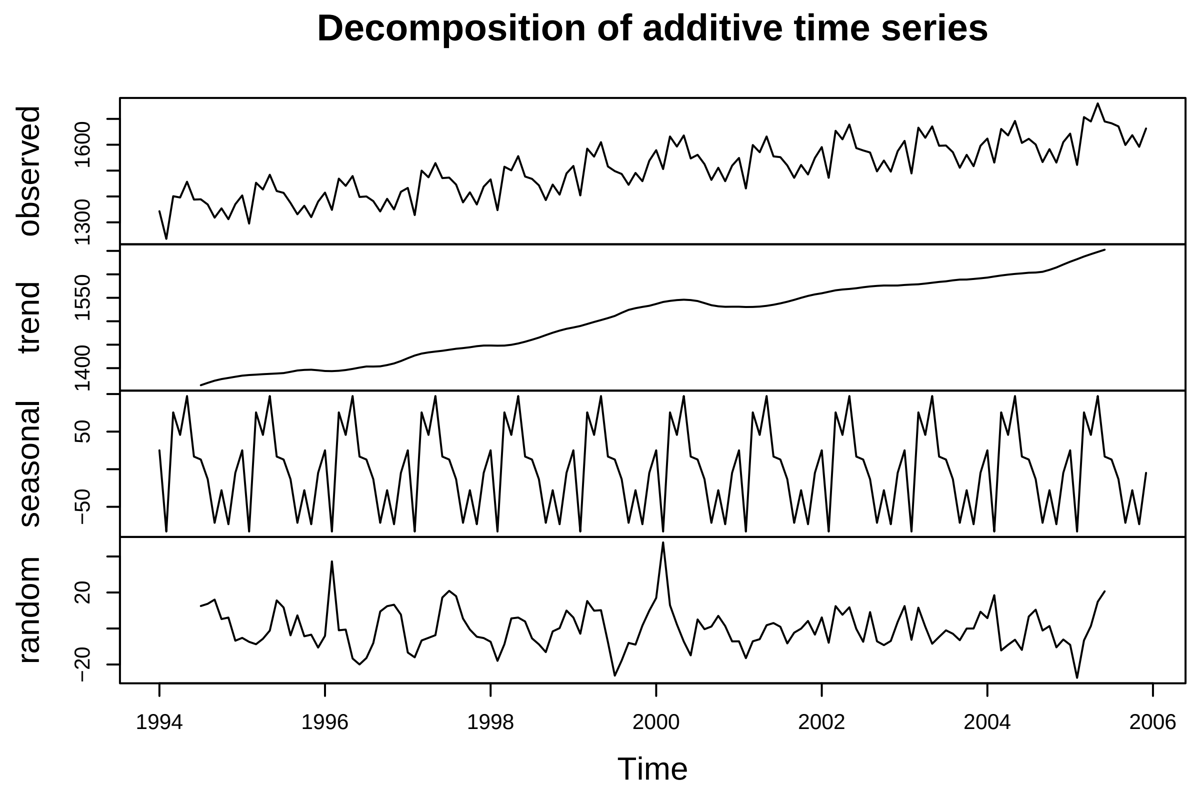
ACF and PACF plots :






```r
# Decompose the Milk Production time series into trend , seasonal , and residual components
decompose_ts(milk_data, "Monthly Milk Production")
```

```
Decomposing the time series for  Monthly Milk Production
```



$x

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1994	1343	1236	1401	1396	1457	1388	1389	1369	1318	1354	1312	1370
1995	1404	1295	1453	1427	1484	1421	1414	1375	1331	1364	1320	1380
1996	1415	1348	1469	1441	1479	1398	1400	1382	1342	1391	1350	1418
1997	1433	1328	1500	1474	1529	1471	1473	1446	1377	1416	1369	1438
1998	1466	1347	1515	1501	1556	1477	1468	1443	1386	1446	1407	1489
1999	1518	1404	1585	1554	1610	1516	1498	1487	1445	1491	1459	1538
2000	1579	1506	1632	1593	1636	1547	1561	1525	1464	1511	1459	1519
2001	1549	1431	1599	1571	1632	1555	1552	1520	1472	1522	1485	1549
2002	1591	1472	1654	1621	1678	1587	1578	1570	1497	1539	1496	1575
2003	1615	1489	1666	1627	1671	1596	1597	1571	1511	1561	1517	1596
2004	1624	1531	1661	1636	1692	1607	1623	1601	1533	1583	1531	1610
2005	1643	1522	1707	1690	1760	1690	1683	1671	1599	1637	1592	1663

$seasonal

	Jan	Feb	Mar	Apr	May	Jun
1994	25.16919	-82.90657	75.61237	45.65783	97.34343	16.80934
1995	25.16919	-82.90657	75.61237	45.65783	97.34343	16.80934
1996	25.16919	-82.90657	75.61237	45.65783	97.34343	16.80934
1997	25.16919	-82.90657	75.61237	45.65783	97.34343	16.80934
1998	25.16919	-82.90657	75.61237	45.65783	97.34343	16.80934

```
1999  25.16919 -82.90657  75.61237  45.65783  97.34343  16.80934
2000  25.16919 -82.90657  75.61237  45.65783  97.34343  16.80934
2001  25.16919 -82.90657  75.61237  45.65783  97.34343  16.80934
2002  25.16919 -82.90657  75.61237  45.65783  97.34343  16.80934
2003  25.16919 -82.90657  75.61237  45.65783  97.34343  16.80934
2004  25.16919 -82.90657  75.61237  45.65783  97.34343  16.80934
2005  25.16919 -82.90657  75.61237  45.65783  97.34343  16.80934
           Jul       Aug       Sep       Oct       Nov       Dec
1994  12.89646 -13.32323 -71.29293 -27.92929 -73.19066  -4.84596
1995  12.89646 -13.32323 -71.29293 -27.92929 -73.19066  -4.84596
1996  12.89646 -13.32323 -71.29293 -27.92929 -73.19066  -4.84596
1997  12.89646 -13.32323 -71.29293 -27.92929 -73.19066  -4.84596
1998  12.89646 -13.32323 -71.29293 -27.92929 -73.19066  -4.84596
1999  12.89646 -13.32323 -71.29293 -27.92929 -73.19066  -4.84596
2000  12.89646 -13.32323 -71.29293 -27.92929 -73.19066  -4.84596
2001  12.89646 -13.32323 -71.29293 -27.92929 -73.19066  -4.84596
2002  12.89646 -13.32323 -71.29293 -27.92929 -73.19066  -4.84596
2003  12.89646 -13.32323 -71.29293 -27.92929 -73.19066  -4.84596
2004  12.89646 -13.32323 -71.29293 -27.92929 -73.19066  -4.84596
2005  12.89646 -13.32323 -71.29293 -27.92929 -73.19066  -4.84596

$trend
          Jan      Feb      Mar      Apr      May      Jun      Jul
1994       NA       NA       NA       NA       NA       NA 1363.625
1995 1384.042 1385.333 1386.125 1387.083 1387.833 1388.583 1389.458
1996 1393.917 1393.625 1394.375 1395.958 1398.333 1401.167 1403.500
1997 1421.208 1426.917 1431.042 1433.542 1435.375 1437.000 1439.208
1998 1448.208 1447.875 1448.125 1449.750 1452.583 1456.292 1460.583
1999 1486.750 1489.833 1494.125 1498.458 1502.500 1506.708 1511.292
2000 1536.875 1541.083 1543.458 1545.083 1545.917 1545.125 1543.083
2001 1530.958 1530.375 1530.500 1531.292 1532.833 1535.167 1538.167
2002 1559.667 1562.833 1565.958 1567.708 1568.875 1570.417 1572.500
2003 1577.375 1578.208 1578.833 1580.333 1582.125 1583.875 1585.125
2004 1593.083 1595.417 1597.583 1599.417 1600.917 1602.083 1603.458
2005 1626.917 1632.333 1638.000 1643.000 1647.792 1652.542       NA
          Aug      Sep      Oct      Nov      Dec
1994 1368.625 1373.250 1376.708 1379.125 1381.625
1995 1392.125 1395.000 1396.250 1396.625 1395.458
1996 1403.417 1403.875 1406.542 1410.000 1415.125
1997 1441.375 1442.792 1444.542 1446.792 1448.167
1998 1465.125 1470.417 1475.542 1480.000 1483.875
```

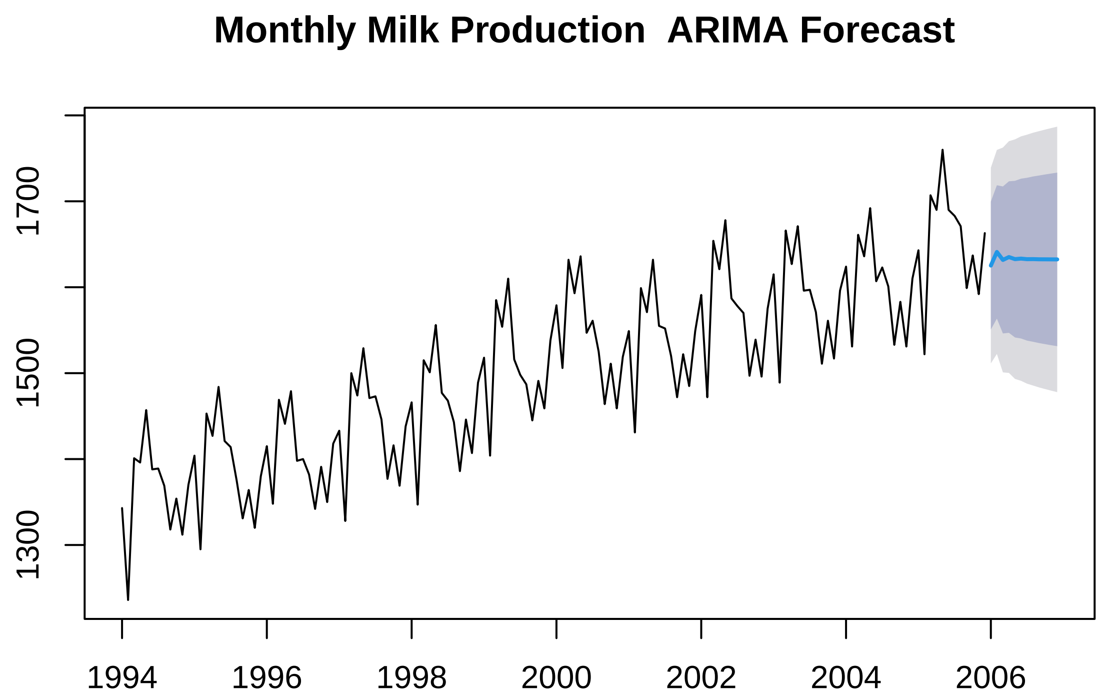
```
1999 1518.083 1524.292 1527.875 1530.583 1532.958
2000 1538.708 1534.208 1531.917 1530.833 1531.000
2001 1541.625 1545.625 1550.000 1554.000 1557.250
2002 1574.208 1575.417 1576.167 1576.125 1576.208
2003 1587.250 1588.792 1588.958 1590.208 1591.542
2004 1603.875 1605.417 1609.583 1614.667 1620.958
2005       NA       NA       NA       NA       NA

$random
             Jan          Feb          Mar          Apr          May
1994          NA           NA           NA           NA           NA
1995  -5.21085859  -7.42676768  -8.73737374  -5.74116162  -1.17676768
1996  -4.08585859  37.28156566  -0.98737374  -0.61616162 -16.67676768
1997 -13.37752525 -16.01010101  -6.65404040  -5.19949495  -3.71843434
1998  -7.37752525 -17.96843434  -8.73737374   5.59217172   6.07323232
1999   6.08080808  -2.92676768  15.26262626   9.88383838  10.15656566
2000  16.95580808  47.82323232  12.92929293   2.25883838  -7.26010101
2001  -7.12752525 -16.46843434  -7.11237374  -5.94949495   1.82323232
2002   6.16414141  -7.92676768  12.42929293   7.63383838  11.78156566
2003  12.45580808  -6.30176768  11.55429293   1.00883838  -8.46843434
2004   5.74747475  18.48989899 -12.19570707  -9.07449495  -6.26010101
2005  -9.08585859 -27.42676768  -6.61237374   1.34217172  14.86489899
             Jun          Jul          Aug          Sep          Oct
1994          NA  12.47853535  13.69823232  16.04292929   5.22095960
1995  15.60732323  11.64520202  -3.80176768   7.29292929  -4.32070707
1996 -19.97601010 -16.39646465  -8.09343434   9.41792929  12.38762626
1997  17.19065657  20.89520202  17.94823232   5.50126263  -0.61237374
1998   3.89898990  -5.47979798  -8.80176768 -13.12373737  -1.61237374
1999  -7.51767677 -26.18813131 -17.76010101  -7.99873737  -8.94570707
2000 -14.93434343   5.02020202  -0.38510101   1.08459596   7.01262626
2001   3.02398990   0.93686869  -8.30176768  -2.33207071  -0.07070707
2002  -0.22601010  -7.39646465   9.11489899  -7.12373737  -9.23737374
2003  -4.68434343  -1.02146465  -2.92676768  -6.49873737  -0.02904040
2004 -11.89267677   6.64520202  10.44823232  -1.12373737   1.34595960
2005  20.64898990           NA           NA           NA           NA
             Nov          Dec
1994   6.06565657  -6.77904040
1995  -3.43434343 -10.61237374
1996  13.19065657   7.72095960
1997  -4.60101010  -5.32070707
1998   0.19065657   9.97095960
```

```
1999   1.60732323   9.88762626
2000   1.35732323  -7.15404040
2001   4.19065657  -3.40404040
2002  -6.93434343   3.63762626
2003  -0.01767677   9.30429293
2004 -10.47601010  -6.11237374
2005           NA           NA

$figure
 [1]  25.16919 -82.90657  75.61237  45.65783  97.34343  16.80934
 [7]  12.89646 -13.32323 -71.29293 -27.92929 -73.19066  -4.84596

$type
[1] "additive"

attr(,"class")
[1] "decomposed.ts"
```

```r
# Fit ARIMA model for the Milk Production dataset
arima_milk <- fit_arima(milk_data, "Monthly Milk Production")
```

```
 Fitting ARIMA model for  Monthly Milk Production

Warning in adf.test(ts_data, alternative = "stationary"): p-value
smaller than printed p-value

ADF Test p- value : 0.01
Series: ts_data
ARIMA(2,1,1)

Coefficients:
         ar1     ar2      ma1
      0.2066  0.3330  -0.9109
s.e.  0.0879  0.0869   0.0336

sigma^2 = 3373:  log likelihood = -782.65
AIC=1573.29   AICc=1573.58   BIC=1585.14

Training set error measures:
                   ME     RMSE      MAE       MPE     MAPE     MASE
Training set 10.57461 57.26184 47.71957 0.5838515 3.166587 1.536337
```

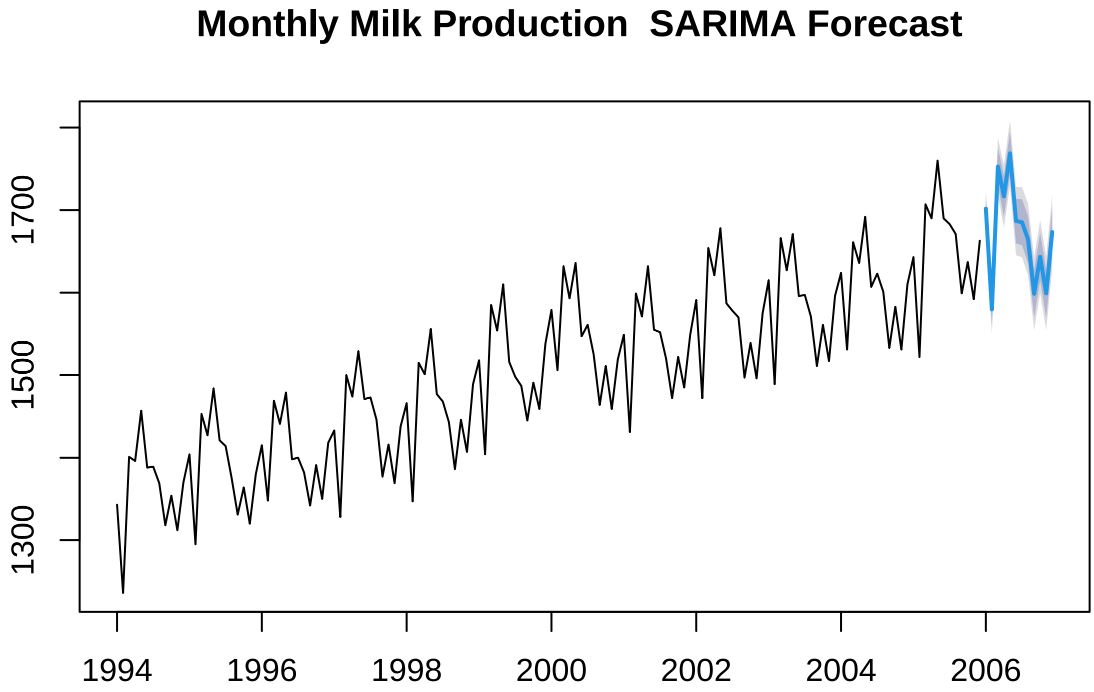
```
                  ACF1
Training set 0.002171886
```




```r
# Fit SARIMA model for the Milk Production dataset
sarima_milk <- fit_sarima(milk_data, "Monthly Milk Production")
```

```
 Fitting SARIMA model for  Monthly Milk Production
Series: ts_data
ARIMA(1,0,0)(2,1,2)[12] with drift

Coefficients:
         ar1    sar1     sar2     sma1    sma2   drift
      0.8638  0.0607  -0.4074  -1.0121  0.4831  2.1882
s.e.  0.0475  0.1862   0.1173   0.1994  0.1881  0.2174

sigma^2 = 137.9:  log likelihood = -518.84
AIC=1051.67   AICc=1052.57   BIC=1071.85

Training set error measures:
                     ME     RMSE      MAE         MPE      MAPE
Training set -0.1211196 10.98512 8.342375 -0.01115387 0.5520753
                  MASE        ACF1
```

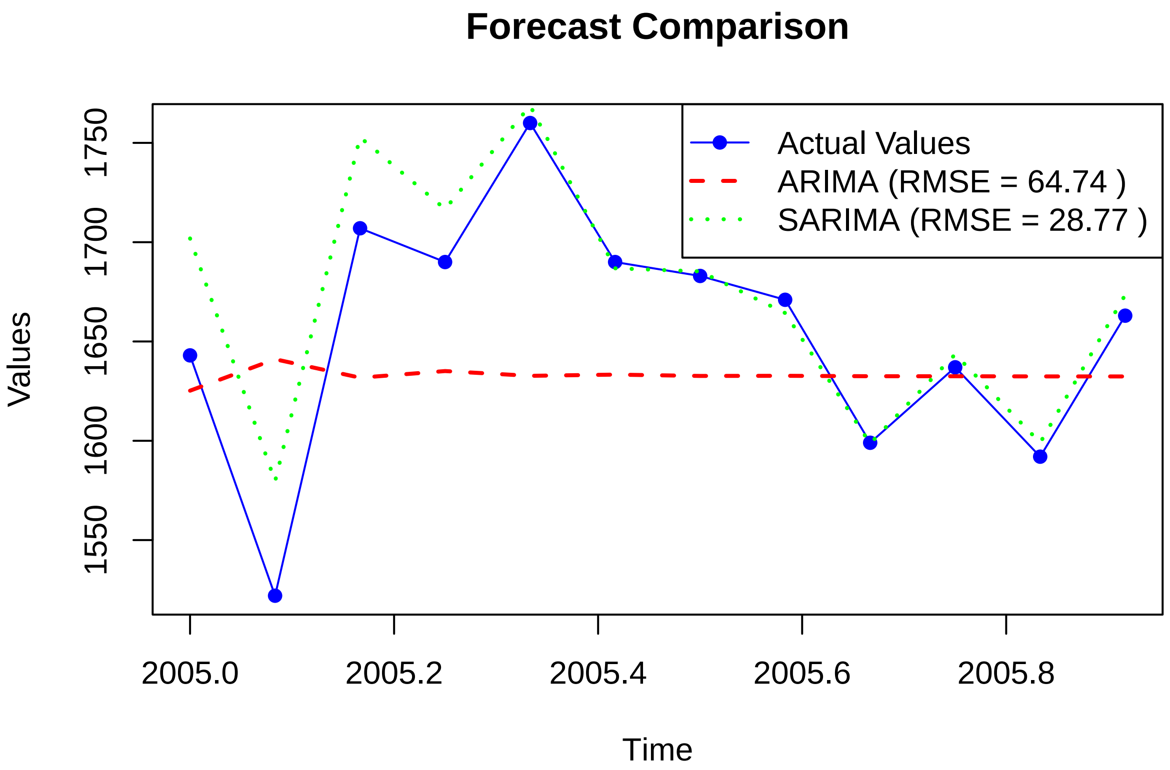
```
Training set 0.2685838 -0.08850265
```




```r
# Compare ARIMA and SARIMA models based on their forecast accuracy
compare_models(arima_milk, sarima_milk, milk_data)
```

```
Comparing ARIMA and SARIMA models :
ARIMA Forecast Accuracy :
21.88452 64.7399 54.04997 1.188661 3.264899 SARIMA Forecast Accuracy :
-17.81697 28.77242 19.52245 -1.093511 1.195402
```

```r
# Forecasting and plot comparison for Milk Production dataset
h_milk <- 12 # Define forecast horizon for Milk Production dataset (12 months ahead )
# Extract the actual values for the last 12 months of the Milk Production data
milk_actual_values <- milk_data[(length(milk_data) - h_milk + 1): length(milk_data)]
# Generate ARIMA forecast for the next 12 months
arima_milk_forecast <- forecast(arima_milk, h = h_milk)$mean
# Generate SARIMA forecast for the next 12 months
sarima_milk_forecast <- forecast(sarima_milk, h = h_milk)$mean
# Extract the time points for the last 12 months
time_points_milk <- time(milk_data)[(length(milk_data) - h_milk + 1): length(milk_data)]
# Plot and compare the forecasts from ARIMA and SARIMA models against the actual values
plot_forecast_comparison(milk_actual_values, arima_milk_forecast, sarima_milk_forecast,
                         time_points_milk)
```



Forecast Comparison
Actual Values
ARIMA (RMSE = 64.74)
SARIMA (RMSE = 28.77)
Values
1550
1600
1650
1700
1750
2005.0
2005.2
2005.4
2005.6
2005.8
Time

Program - 10

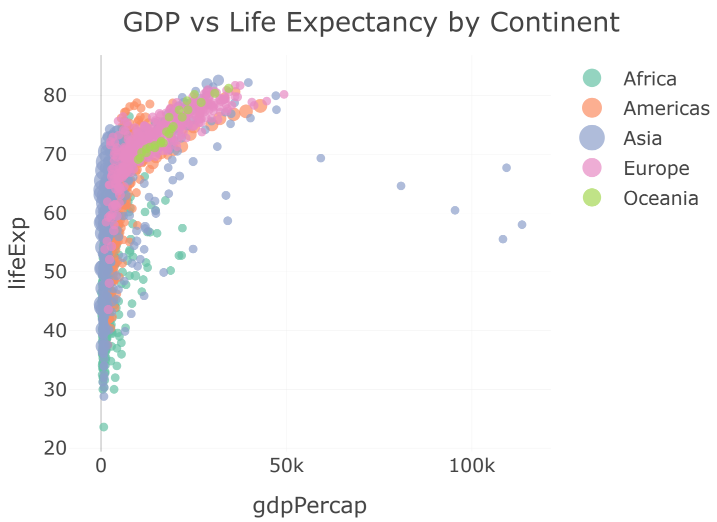
Interactive Visualization with plotly and Dynamic Reports with RMark-down

Date of Execution - 2025-10-28

Objective - This program tests students' abilities to create interactive visualizations and generate dynamic reports using plotly and RMarkdown.

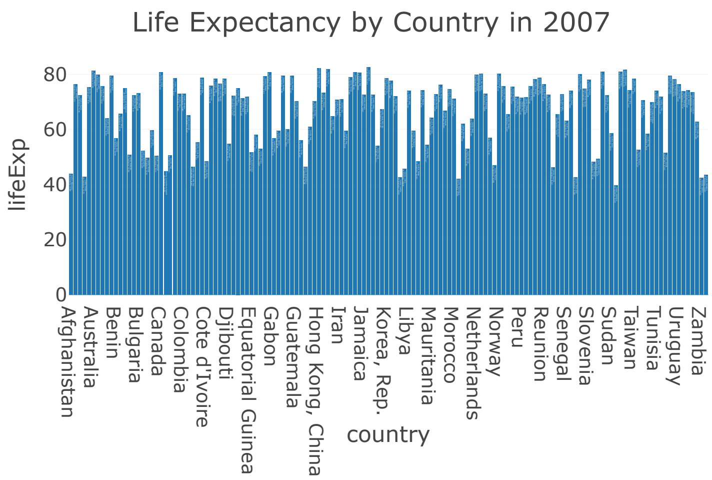
```r
# Load necessary libraries
library(plotly)
library(gapminder)
library(dplyr)

# Load Gapminder dataset
data("gapminder")

# Scatter plot with plotly
# Scatter plot of GDP vs Life Expectancy by Continent
scatter_plot <- gapminder %>%
  plot_ly(x = ~gdpPercap, y = ~lifeExp, color = ~continent, size = ~pop,
          hoverinfo = 'text', text = ~paste("Country:", country, "<br>GDP per Capita:", gdpPercap),
          type = 'scatter', mode = 'markers') %>%
  layout(title = 'GDP vs Life Expectancy by Continent',
         margin = list(l = 20, r = 20, b = 20, t = 30)
  )

# Display the scatter plot
scatter_plot
```

```
Warning: 'line.width' does not currently support multiple values.
Warning: 'line.width' does not currently support multiple values.
Warning: 'line.width' does not currently support multiple values.
Warning: 'line.width' does not currently support multiple values.
Warning: 'line.width' does not currently support multiple values.
```




```
# Bar chart with plotly
# Filter for year 2007 and create a bar chart of life expectancy by country
bar_chart <- gapminder %>%
  filter(year == 2007) %>%
  plot_ly(x = ~country, y = ~lifeExp, type = 'bar',
```

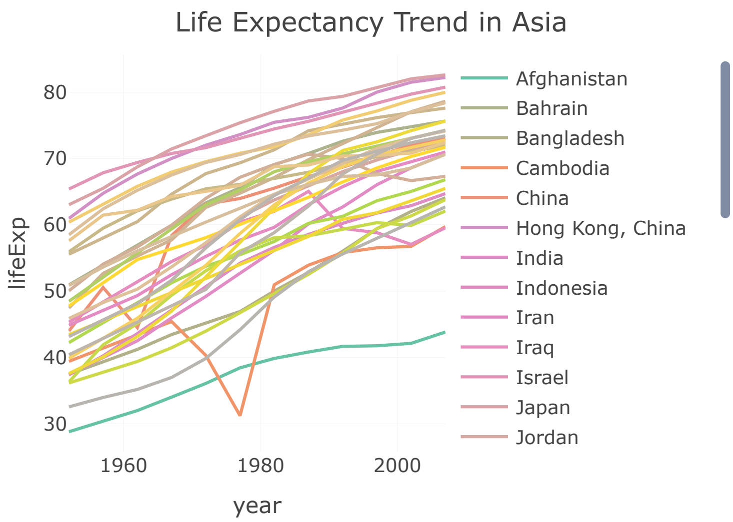
```
          hoverinfo = 'text', text = ~paste("Country:", country, "<br>Life Expectancy:", lifeExp)) %>%
  layout(title = 'Life Expectancy by Country in 2007',
         margin = list(l = 20, r = 20, b = 20, t = 30)
  )

# Display the bar chart
bar_chart
```




```r
# Line chart with plotly
# Filter data for Asia and create a line chart showing life expectancy trends over time
line_chart <- gapminder %>%
  filter(continent == 'Asia') %>%
  plot_ly(x = ~year, y = ~lifeExp, color = ~country, type = 'scatter', mode = 'lines') %>%
```

```r
  layout(title = 'Life Expectancy Trend in Asia',
         margin = list(l = 20, r = 20, b = 20, t = 30)
  )

# Display the line chart
line_chart
```

```
Warning in RColorBrewer::brewer.pal(max(N, 3L), "Set2"): n too large, allowed maximum for palette Set2
Returning the palette you asked for with that many colors

Warning in RColorBrewer::brewer.pal(max(N, 3L), "Set2"): n too large, allowed maximum for palette Set2
Returning the palette you asked for with that many colors
```

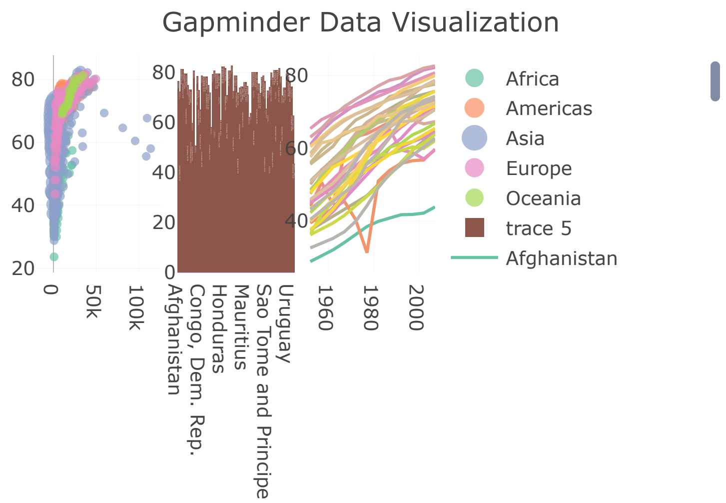



```r
# Combine the Plots
# Combine the scatter, bar, and line charts into one interactive layout
dashboard <- subplot(scatter_plot, bar_chart, line_chart, nrows = 1) %>%
  layout(title = 'Gapminder Data Visualization',
         margin = list(l = 20, r = 20, b = 20, t = 30))
```

```
)
```

```
Warning: ‘line.width‘ does not currently support multiple values.

Warning: ‘line.width‘ does not currently support multiple values.
Warning: ‘line.width‘ does not currently support multiple values.
Warning: ‘line.width‘ does not currently support multiple values.
Warning: ‘line.width‘ does not currently support multiple values.

Warning in RColorBrewer::brewer.pal(max(N, 3L), "Set2"): n too large, allowed maximum for palette Set2
Returning the palette you asked for with that many colors
Warning in RColorBrewer::brewer.pal(max(N, 3L), "Set2"): n too large, allowed maximum for palette Set2
Returning the palette you asked for with that many colors
```

```
# Display the dashboard
dashboard
```


Gapminder Data Visualization
Africa
Americas
Asia
Europe
Oceania
trace 5
Afghanistan
80
60
40
20
0
50k
100k
Afghanistan
Congo, Dem. Rep.
Honduras
Mauritius
Sao Tome and Principe
Uruguay
1960
1980
2000

B.M.S. College of Engineering

Dept. of CSE (Data Science)

Basavanagudi, Bangalore-19