CANARA ENGINEERING COLLEGE

Benjanapadavu – 574219, MANGALORE
Affiliated to Visvesvaraya Technological University

DEPARTMENT OF

COMPUTER SCIENCE AND BUSINESS SYSTEM

COMPUTATIONAL STATISTICS LAB LABORATORY MANUAL

(BCBL504)

V SEMESTER

Mrs.Pooja Kini, Assistant Professor ,CSBS

VISION

The Department of Information Science and Engineering strives to be a centre of learning in the field of Information Technology to produce globally competent engineers catering to the needs of the industry and society.

MISSION

- Impart technical skills in the field of Information Science & Engineering.
- Train and transform students to become technological thinkers and facilitatea quality venture which meets the industrial and societal needs.
- Encourage students to become well-rounded in their professionalcompetencies.

PROGRAM EDUCATIONAL OBJECTIVES

1. Graduates will succeed in the field of Computer Science and Business System, professional career and higher studies.
2. Graduates will analyze the requirements of the software industries and provide novel engineering designs and efficient solutions with legal and ethical responsibility.
3. Graduates will adapt to emerging technologies, work in multidisciplinary teams with effective communication skills and leadership qualities.

PROGRAM OUTCOMES

Engineering graduates in Computer Science and Business System will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

1. An ability to understand, analyze and impart the basic knowledge of Information Science and Engineering.
2. An ability to apply the programming, designing, and problem solving techniques in building/simulating the applications, solving the problems and guiding the innovative career paths to become an IT Engineer.

COMPUTATIONAL STATISTICS LAB		Semester	**V**
Course Code	**BCBL504**	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	0:0:2:0	SEE Marks	50
Credits	01	Exam Hours	100
Examination type (SEE)	Practical		

Course objectives:

- To understand the mean, variance, regression models and error term for use in Multivariate data analysis.
- To understand the correlation between the data for decision making.
- To understand the various tests used for the data analysis.
- To explore various techniques for data analysis and visualize the results.

Sl.NO	Experiments (Implementation using Python/R Programming)
1	Program on data wrangling: Combining and merging datasets, Reshaping and Pivoting
2	Program on Data Transformation: String Manipulation, Regular Expressions
3	Program on Time series: GroupBy Mechanics to display in data vector, multivariate time series and forecasting formats
4	Program to measure central tendency and measures of dispersion: Mean, Median, Mode, Standard Deviation, Variance, Mean deviation and Quartile deviation for a frequency distribution/data.
5	Program to perform cross validation for a given dataset to measure Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and R^2 Error using Validation Set, Leave One Out Cross-Validation(LOOCV) and K-fold Cross-Validation approaches
6	Program to display Normal, Binomial Poisson, Bernoulli disrtibutions for a given frequency distribution and analyze the results.
7	Program to implement one sample, two sample and paired sample t-tests for a sample data and analyse the results.
8	Program to implement One-way and Two-way ANOVA tests and analyze the results
9	Program to implement correlation, rank correlation and regression and plot x-y plot and heat maps of correlation matrices.
10	Program to implement PCA for Wisconsin dataset, visualize and analyze the results.
11	Program to implement the working of linear discriminant analysis using iris dataset and visualize the results.
12	Program to Implement multiple linear regression using iris dataset, visualize and analyze the results.

Course outcomes (Course Skill Set):
At the end of the course the student will be able to:

- Design the experiment for the given problem using statistical methods.
- Develop the solution for the given real world problem using statistical techniques.
- Analyze the results and produce substantial written documentation.

Introduction to R programming:

R is a programming language and free software developed by Ross Ihaka and Robert Gentleman in 1993. R possesses an extensive catalog of statistical and graphical methods. It includes machine learning algorithms, linear regression, time series, statistical inference to name a few. Most of the R libraries are written in R, but for heavy computational tasks, C, C++ and Fortran codes are preferred. R is not only entrusted by academic, but many large companies also use R programming language, including Uber, Google, Airbnb, Facebook and so on.

Data analysis with R is done in a series of steps; programming, transforming, discovering, modeling and communicate the results.

Program: R is a clear and accessible programming tool

Transform: R is made up of a collection of libraries designed specifically for data science

Discover: Investigate the data, refine your hypothesis and analyze them

Model: R provides a wide array of tools to capture the right model for your data

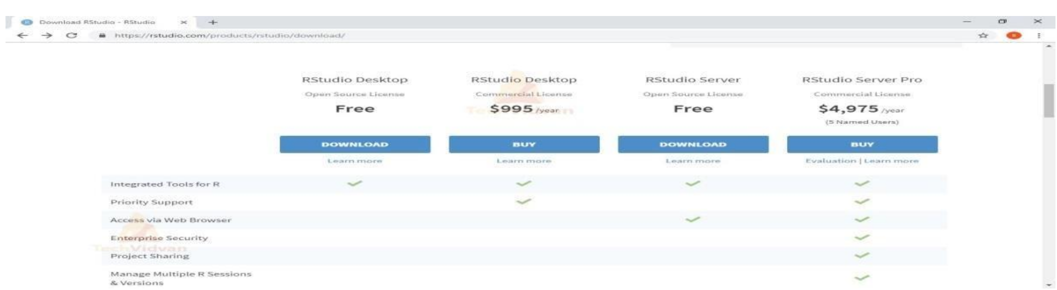
Communicate: Integrate codes, graphs, and outputs to a report with R Markdown or build Shiny apps to share with the world

What is R used for?

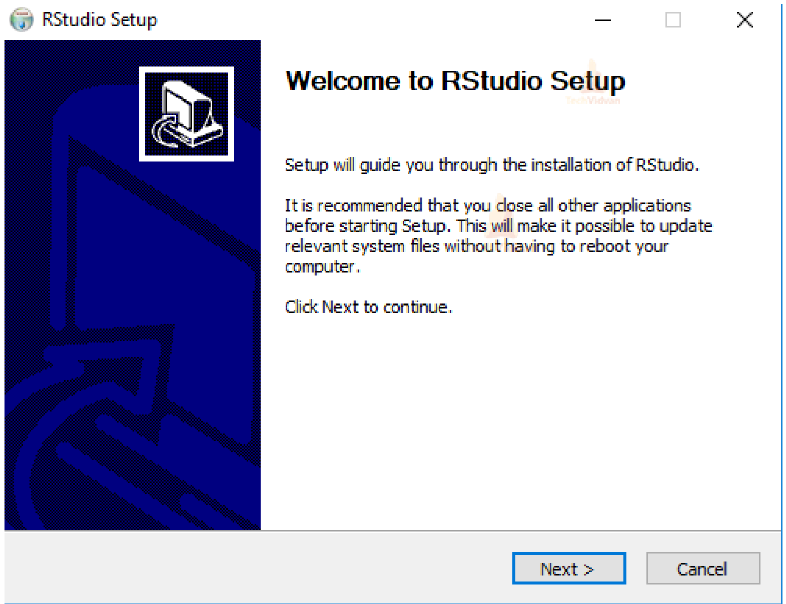
- Statistical inference
- Data analysis
- Machine learning algorithm

Installation of R-Studio on windows:

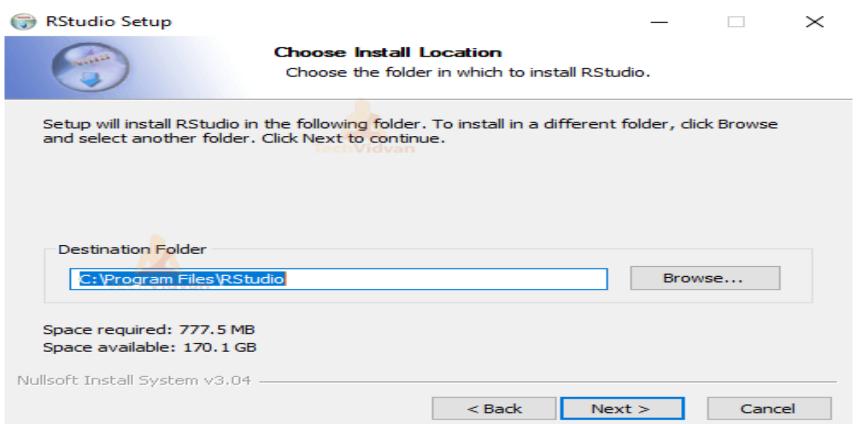
Step – 1: With R-base installed, let's move on to installing RStudio. To begin, goto download RStudio and click on the download button for RStudio desktop.



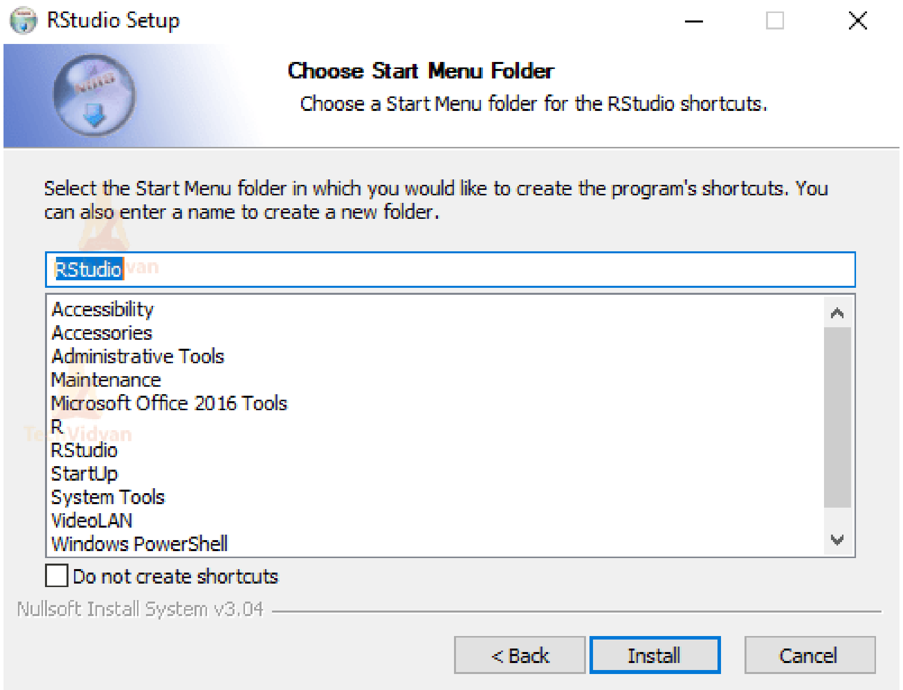
- Step – 2: Click on the link for the windows version of RStudio and save the .exe file.
- Step – 3: Run the .exe and follow the installation instructions.
- 3. Click Next on the welcome window.



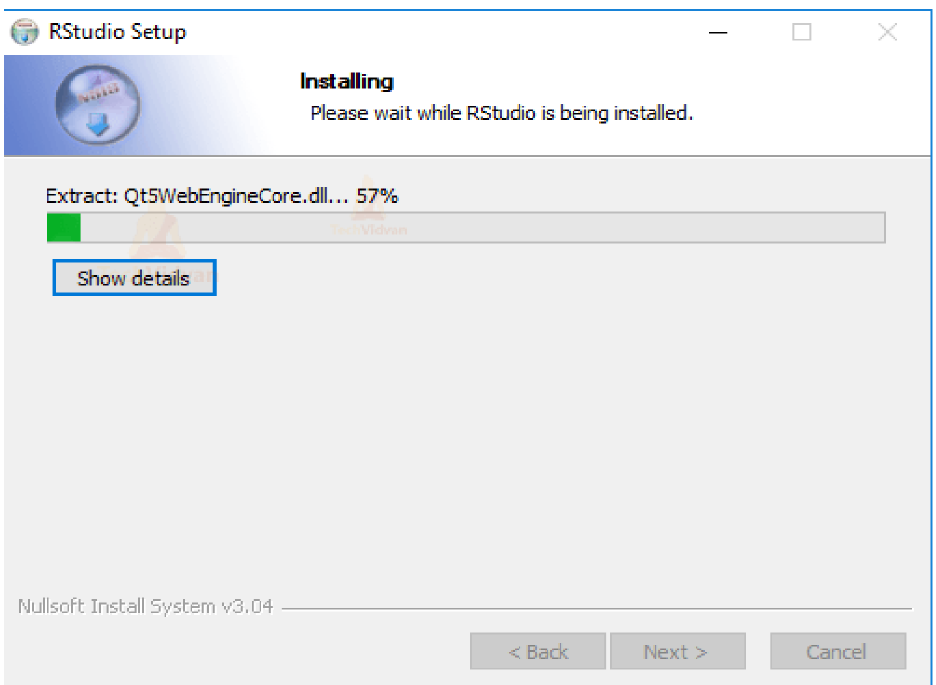
- Enter/browse the path to the installation folder and click Next to proceed.



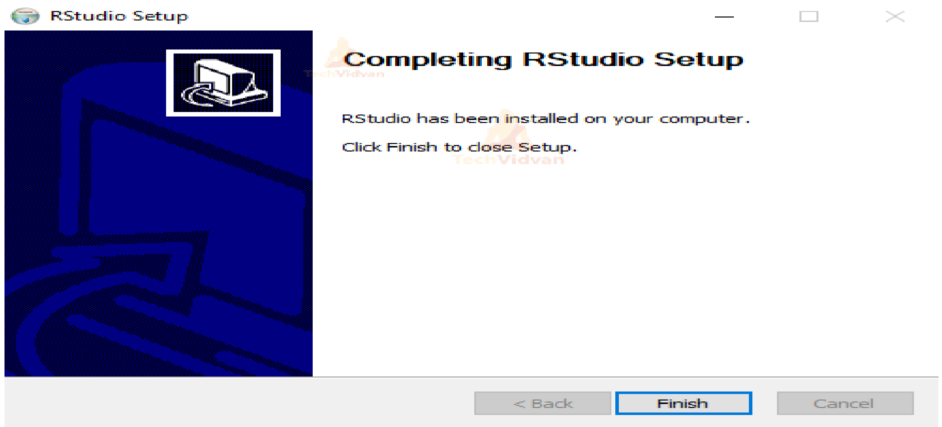
Select the folder for the start menu shortcut or click on do not create shortcuts and thenclick Next.



Wait for the installation process to complete.



Click Finish to end the installation.



Install the R Packages:-

In R Studio, if you require a particular library, then you can go through the following instructions:

- First, run R Studio.
- After clicking on the packages tab, click on install. The following dialog box will appear.
- In the Install Packages dialog, write the package name you want to install under the Packages field and then click install. This will install the package you searched for or give you a list of matching packages based on your package text.

Installing Packages:-

The most common place to get packages from is CRAN. To install packages from CRAN you use install. packages("package name"). For instance, if you want to install the ggplot2 package, which is a very popular visualization package, you would type the following in the console:-

Syntax:-

install package from CRAN
install .packages("ggplot2")

Loading Packages:-

Once the package is downloaded to your computer you can access the functions and resources provided by the package in two different ways:

# load the package to use in the current R session
library(packagename)

Getting Help on Packages:-

For more direct help on packages that are installed on your computer you can use the help and vignette functions. Here we can get help on the ggplot2 package with the following:

help(package = "ggplot2") # provides details regarding contents of a package
vignette(package = "ggplot2") # list vignettes available for a specific package
vignette("ggplot2-specs") # view specific vignette
vignette() # view all vignettes on your computer

Experiment 01: Program on data wrangling: Combining and merging datasets, Reshaping and Pivoting

```
# Combining and Merging datasets
df1 <- data.frame(ID = 1:5, Name = c("A", "B", "C", "D", "E"))
df2 <- data.frame(ID = 3:7, Age = c(25, 28, 22, 30, 26))


# Merging on ID
merged_df<- merge(df1, df2, by = "ID", all = TRUE)
print(merged_df)


# Reshaping data (pivoting)
library(tidyr)
long_df<- gather(merged_df, key = "Variable", value = "Value", -ID)
print(long_df)


# Pivoting back to wide format
wide_df<- spread(long_df, key = "Variable", value = "Value")
print(wide_df)
```

OUTPUT:

```
[Workspace loaded from ~/.RData]

 > # Combining and Merging datasets
 > df1 <- data.frame(ID = 1:5, Name = c("A", "B", "C", "D", "E"))
 > df2 <- data.frame(ID = 3:7, Age = c(25, 28, 22, 30, 26))

 > # Merging on ID
 > merged_df<- merge(df1, df2, by = "ID", all = TRUE)
 > print(merged_df)
   ID Name Age
 1  1    A  NA
 2  2    B  NA
 3  3    C  25
 4  4    D  28
 5  5    E  22
 6  6 <NA>  30
 7  7 <NA>  26
 >
 > # Reshaping data (pivoting)
> library(tidyr)
```

```
> print(long_df)
   ID Variable Value
1   1     Name     A
2   2     Name     B
3   3     Name     C
4   4     Name     D
5   5     Name     E
6   6     Name  <NA>
7   7     Name  <NA>
8   1      Age  <NA>
9   2      Age  <NA>
10  3      Age    25
11  4      Age    28
12  5      Age    22
13  6      Age    30
14  7      Age    26
>
>
```

```
> print(wide_df)
  ID  Age Name
1  1 <NA>    A
2  2 <NA>    B
3  3   25    C
4  4   28    D
5  5   22    E
6  6   30 <NA>
7  7   26 <NA>
> 
```

Experiment 02: Program on Data Transformation: String Manipulation, Regular Expressions

```
# String Manipulation
strings<- c("Data_Analysis", "Computational_Statistics")
library(stringr)


# Replace underscores with spaces
strings<- str_replace_all(strings, "_", " ")
print(strings)


# Regular Expressions
emails<- c("user1@example.com", "user2@gmail.com")
valid_emails<- grep("@example.com", emails, value = TRUE)
print(valid_emails)
```

OUTPUT:

```
> # Replace underscores with spaces
> strings<- str_replace_all(strings, "_", " ")
> print(strings)
[1] "Data Analysis"            "Computational Statistics"
>
>
> # Regular Expressions
> emails<- c("user1@example.com", "user2@gmail.com")
> valid_emails<- grep("@example.com", emails, value = TRUE)
> print(valid_emails)
[1] "user1@example.com"
>
```

Experiment 03: Program on Time series: GroupBy Mechanics to display in data vector, multivariate time series andforecasting formats

```
# Load required libraries
library(dplyr)
library(lubridate)
library(forecast)
# Sample data creation
set.seed(123)

date_seq <- seq(from = as.Date("2020-01-01"), by = "month", length.out = 36)
data <- data.frame(
Date = date_seq,
Value1 = rnorm(36, mean = 200, sd = 20),
Value2 = rnorm(36, mean = 100, sd = 10)
)
# Display the original data
print("Original Data:")
print(data)

# Convert to time series
data_ts <- ts(data[,-1], frequency = 12, start = c(2020, 1))

# Grouping by year and calculating mean for each variable
grouped_data <- data %>%
mutate(Year = year(Date)) %>%
group_by(Year) %>%
summarise(Mean_Value1 = mean(Value1),
Mean_Value2 = mean(Value2))

# Display grouped data
print("Grouped Data:")
print(grouped_data)

# Multivariate time series handling
# This creates a time series object for both Value1 and Value2
multivariate_ts <- ts(data[, -1], start = c(2020, 1), frequency = 12)

# Forecasting with ARIMA for Value1
fit_value1 <- auto.arima(multivariate_ts[, 1])
forecast_value1 <- forecast(fit_value1, h = 6) # Forecasting for next 6 months

# Forecasting with ARIMA for Value2
fit_value2 <- auto.arima(multivariate_ts[, 2])
forecast_value2 <- forecast(fit_value2, h =6) # Forecasting for next 6 months
```

```
# Plot the forecasts
par(mfrow = c(2, 1)) # Arrange plots vertically
plot(forecast_value1, main = "Forecast for Value1", xlab = "Time", ylab = "Value1")
plot(forecast_value2, main = "Forecast for Value2", xlab = "Time", ylab = "Value2")

# Reset plot layout
par(mfrow = c(1,2))
print(forecast_value1)
print(forecast_value2)
checkresiduals(fit_value1)
```

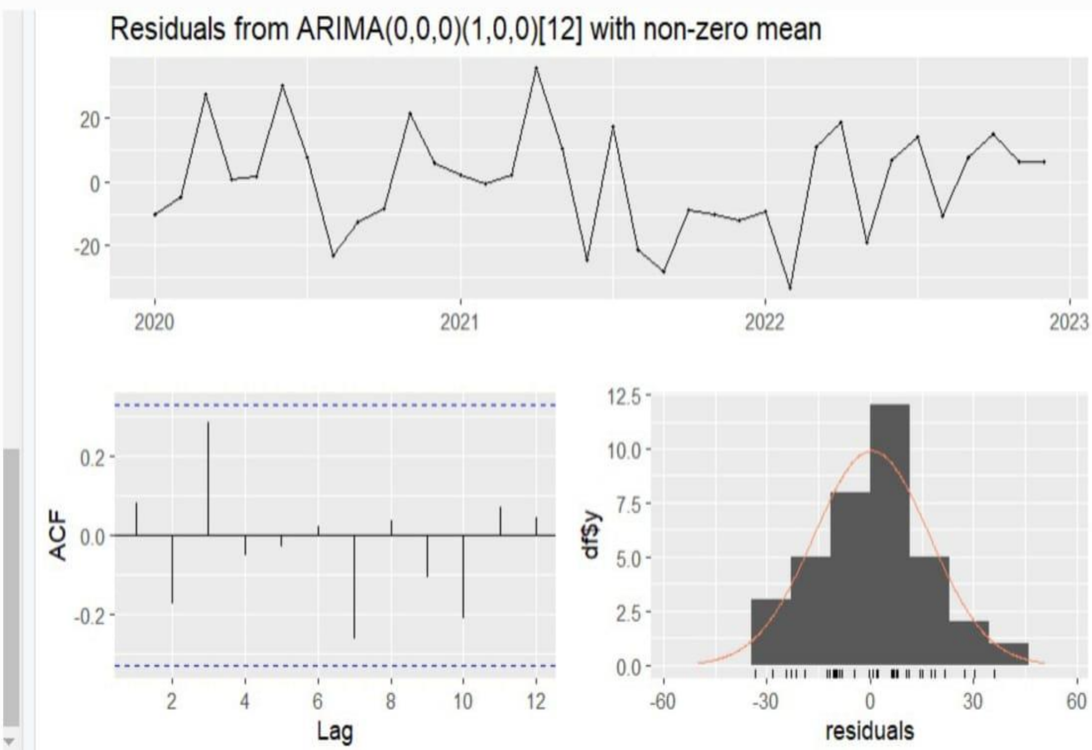
OUTPUT:

```
[1] "Original Data:"
> print(data)
         Date   Value1    Value2
1  2020-01-01 188.7905 105.53918
2  2020-02-01 195.3965  99.38088
3  2020-03-01 231.1742  96.94037
4  2020-04-01 201.4102  96.19529
5  2020-05-01 202.5858  93.05293
6  2020-06-01 234.3013  97.92083
7  2020-07-01 209.2183  87.34604
8  2020-08-01 174.6988 121.68956
9  2020-09-01 186.2629 112.07962
10 2020-10-01 191.0868  88.76891
11 2020-11-01 224.4816  95.97115
12 2020-12-01 207.1963  95.33345
13 2021-01-01 208.0154 107.79965
14 2021-02-01 202.2137  99.16631
15 2021-03-01 188.8832 102.53319
16 2021-04-01 235.7383  99.71453
17 2021-05-01 209.9570  99.57130
18 2021-06-01 160.6677 113.68602
19 2021-07-01 214.0271  97.74229
20 2021-08-01 190.5442 115.16471
21 2021-09-01 178.6435  84.51247
22 2021-10-01 195.6405 105.84614
23 2021-11-01 179.4799 101.23854
24 2021-12-01 185.4222 102.15942
25 2022-01-01 187.4992 103.79639
26 2022-02-01 166.2661  94.97677
27 2022-03-01 216.7557  96.66793
28 2022-04-01 203.0675  89.81425
29 2022-05-01 177.2373  89.28209
30 2022-06-01 225.0763 103.03529
31 2022-07-01 208.5293 104.48210
32 2022-08-01 194.0986 100.53004
33 2022-09-01 217.9025 109.22267
34 2022-10-01 217.5627 120.50085
35 2022-11-01 216.4316  95.08969
36 2022-12-01 213.7728  76.90831
```

```
[1] "Grouped Data:"
> print(grouped_data)
# A tibble: 3 × 3
   Year Mean_Value1 Mean_Value2
  <dbl>       <dbl>       <dbl>
1  2020        204.        99.2
2  2021        196.       102. 
3  2022        204.        98.7
```

GRAPH:

```
> print(forecast_value1)
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Jan 2023       206.2451 184.5284 227.9619 173.0323 239.4580
Feb 2023       215.8344 194.1176 237.5511 182.6215 249.0473
Mar 2023       193.0324 171.3156 214.7491 159.8195 226.2453
Apr 2023       199.2142 177.4975 220.9310 166.0014 232.4271
May 2023       210.8796 189.1629 232.5964 177.6667 244.0925
Jun 2023       189.2747 167.5579 210.9914 156.0618 222.4875
> print(forecast_value2)
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Jan 2023       100.1016 87.88564 112.3176 81.41889 118.7844
Feb 2023       100.1016 87.88564 112.3176 81.41889 118.7844
Mar 2023       100.1016 87.88564 112.3176 81.41889 118.7844
Apr 2023       100.1016 87.88564 112.3176 81.41889 118.7844
May 2023       100.1016 87.88564 112.3176 81.41889 118.7844
Jun 2023       100.1016 87.88564 112.3176 81.41889 118.7844
> checkresiduals(fit_value1)

	Ljung-Box test

data:  Residuals from ARIMA(0,0,0)(1,0,0)[12] with non-zero mean
Q* = 8.2274, df = 6, p-value = 0.2219

Model df: 1.   Total lags used: 7
```



Experiment 04: Program to measure central tendency and measures of dispersion: Mean, Median, Mode, Standard Deviation, Variance, Mean deviation and Quartile deviation for a frequency distribution/data.

```
data<- c(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)

# Central Tendency
mean_value<- mean(data)
median_value<- median(data)
mode_value<- as.numeric(names(sort(table(data), decreasing=TRUE)[1]))


# Dispersion Measures
sd_value<- sd(data)
var_value<- var(data)
mad_value<- mad(data)
quartile_deviation<- IQR(data) / 2


list(mean = mean_value, median = median_value, mode = mode_value,
sd = sd_value, variance = var_value, MAD = mad_value, quartile_dev =
quartile_deviation)
```

OUTPUT:

```
$mean
[1] 55

$median
[1] 55

$mode
[1] 10

$sd
[1] 30.2765

$variance
[1] 916.6667

$MAD
[1] 37.065

$quartile_dev
[1] 22.5

>
```

Experiment 05: Program to perform cross validation for a given dataset to measure Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and R^2 Error using Validation Set, Leave One Out Cross-Validation(LOOCV)and K-fold Cross-Validation approaches

```
# Load necessary libraries
library(caret)     # For cross-validation functions
library(Metrics)   # For performance metrics
library(dplyr)     # For data manipulation

# Load dataset
data(mtcars)
# Set the target variable and predictor variables
target_variable<- "mpg"
predictors<- setdiff(names(mtcars), target_variable)


# Split the data into training and validation sets (80% training, 20% validation)
set.seed(123)  # For reproducibility
trainIndex<- createDataPartition(mtcars$mpg, p = 0.8, list = FALSE)
trainData<- mtcars[trainIndex, ]
validData<- mtcars[-trainIndex, ]

# Function to calculate RMSE, MAE, and R2
calc_metrics<- function(actual, predicted) {
rmse<- rmse(actual, predicted)
mae<- mae(actual, predicted)
  r2 <- cor(actual, predicted)^2  # R-squared
return(c(RMSE = rmse, MAE = mae, R2 = r2))
}

# Validation Set Approach
model_val<- lm(mpg ~ ., data = trainData)
pred_val<- predict(model_val, newdata = validData)
metrics_val<- calc_metrics(validData$mpg, pred_val)
print("Validation Set Metrics:")
print(metrics_val)


# Leave-One-Out Cross-Validation (LOOCV)
loocv_metrics<- sapply(1:nrow(mtcars), function(i) {
train_loocv<- mtcars[-i, ]
test_loocv<- mtcars[i, , drop = FALSE]
model_loocv<- lm(mpg ~ ., data = train_loocv)
pred_loocv<- predict(model_loocv, newdata = test_loocv)
```

```
calc_metrics(test_loocv$mpg, pred_loocv)
})
loocv_metrics_avg<- colMeans(loocv_metrics)
print("LOOCV Metrics (Average across folds):")
print(loocv_metrics_avg)


# K-Fold Cross-Validation (5-fold)
k <- 5
cv_results<- train(mpg ~ ., data = mtcars, method = "lm",
trControl = trainControl(method = "cv", number = k,
summaryFunction = defaultSummary))
print("K-Fold Cross-Validation Metrics:")
print(cv_results$results)
```

OUTPUT:

```
[1] "Validation Set Metrics:"
> print(metrics_val)
     RMSE       MAE        R2
4.8089808 3.3848440 0.6297763
>
> # Leave-One-Out Cross-Validation (LOOCV)
> loocv_metrics<- sapply(1:nrow(mtcars), function(i)
+   train_loocv<- mtcars[-i, ]
+   test_loocv<- mtcars[i, , drop = FALSE]
+   model_loocv<- lm(mpg ~ ., data = train_loocv)
+   pred_loocv<- predict(model_loocv, newdata = test_
+   calc_metrics(test_loocv$mpg, pred_loocv)
+ })
> loocv_metrics_avg<- colMeans(loocv_metrics)
> print("LOOCV Metrics (Average across folds):")
[1] "LOOCV Metrics (Average across folds):"
> print(loocv_metrics_avg)
 [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
NA NA NA NA NA NA NA NA NA NA
[31] NA NA
>
> # K-Fold Cross-Validation (5-fold)
> k <- 5
> cv_results<- train(mpg ~ ., data = mtcars, method
+                      trControl = trainControl(method
ber = k,
+                                                  summar
defaultSummary))
> print("K-Fold Cross-Validation Metrics:")
[1] "K-Fold Cross-Validation Metrics:"
> print(cv_results$results)
  intercept     RMSE  Rsquared      MAE   RMSESD Rsqu
D
1      TRUE 4.529418 0.7152401 3.656295 2.17539   0.
4
```

Experiment 06: Program to display Normal, Binomial Poisson, Bernoulli disrtibutions for a given frequency distribution and analyze the results.

```
# Load required libraries
if (!requireNamespace("ggplot2", quietly = TRUE)) install.packages("ggplot2")
if (!requireNamespace("dplyr", quietly = TRUE)) install.packages("dplyr")

library(ggplot2)
library(dplyr)

# Set seed for reproducibility
set.seed(123)

# Create a frequency distribution as a data frame
# Example: Frequency distribution of a survey response
response_data <- data.frame(
  response = c("Strongly Disagree", "Disagree", "Neutral", "Agree", "Strongly Agree"),
  frequency = c(5, 15, 25, 30, 25)
)

# Display the frequency distribution
print("Frequency Distribution:")
print(response_data)

# Calculate proportions
response_data <- response_data %>%
  mutate(proportion = frequency / sum(frequency))

# Normal Distribution parameters
mu <- mean(1:length(response_data$response)) # Mean of the response categories
sigma <- sd(1:length(response_data$response)) # Standard deviation

# Binomial Distribution parameters
n <- sum(response_data$frequency)  # Total number of responses
p <- mean(response_data$proportion) # Probability of success


# Poisson Distribution parameter (lambda)
lambda <- n * p

# Bernoulli Distribution probabilities
p_bernoulli <- response_data$proportion[1]  # Probability of "Strongly Disagree"
```

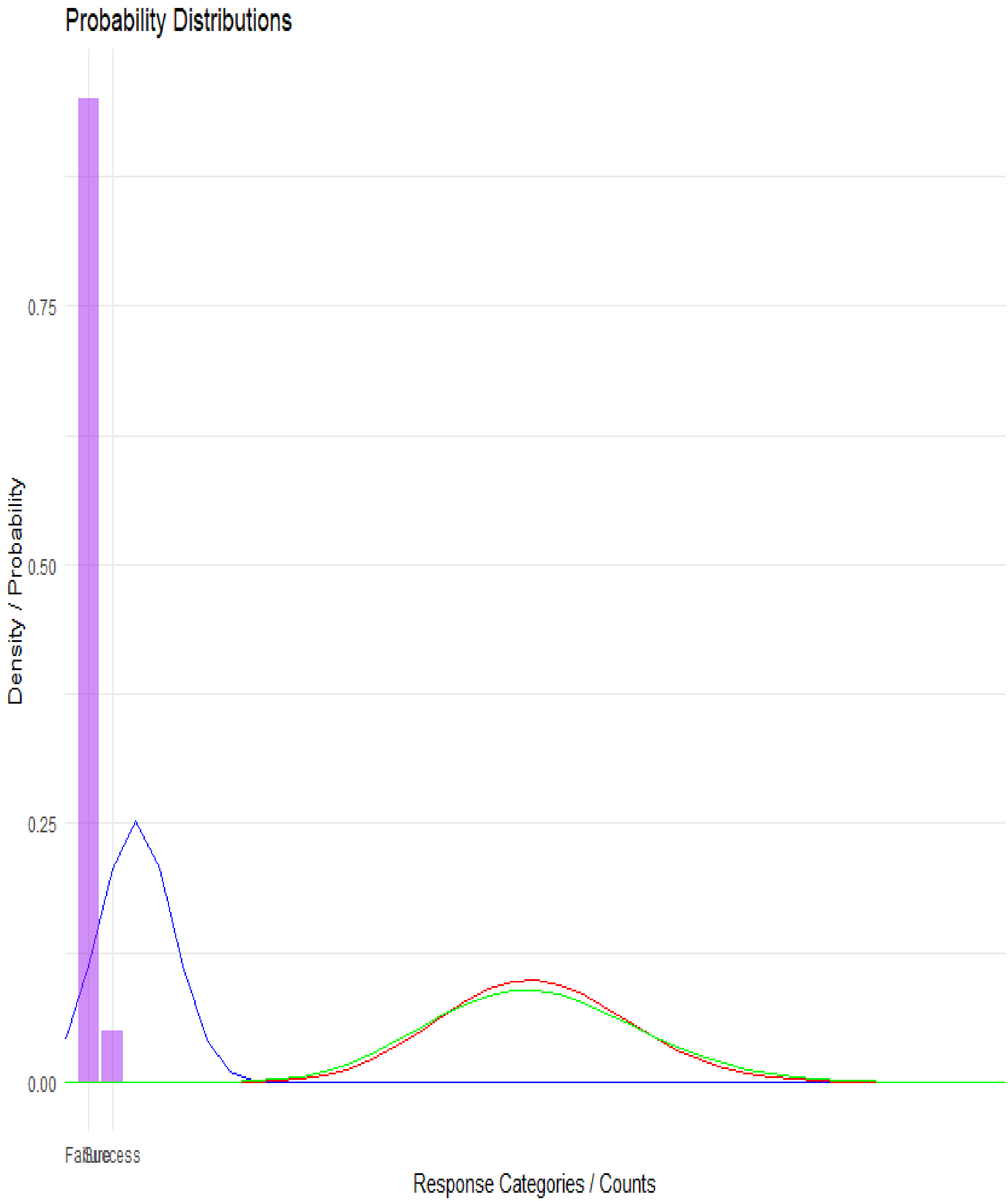
```
# Generate values for distributions
x <- 0:(max(response_data$frequency) + 10)

# Create data frame for Normal distribution
normal_df <- data.frame(
  x = x,
  density = dnorm(x, mean = mu, sd = sigma)
)

# Create data frame for Binomial distribution
binomial_df <- data.frame(
  x = x,
  density = dbinom(x, size = n, prob = p)
)

# Create data frame for Poisson distribution
poisson_df <- data.frame(
  x = x,
  density = dpois(x, lambda = lambda)
)


# Create data frame for Bernoulli distribution
bernoulli_df <- data.frame(
  x = c(0, 1),
  density = c(1 - p_bernoulli, p_bernoulli) # 0 for failure, 1 for success
)


# Visualize distributions
ggplot() +
  geom_line(data = normal_df, aes(x = x, y = density), color = "blue") +
  geom_line(data = binomial_df, aes(x = x, y = density), color = "red") +
  geom_line(data = poisson_df, aes(x = x, y = density), color = "green") +
  geom_bar(data = bernoulli_df, aes(x = factor(x), y = density), stat = "identity", fill =
"purple", alpha = 0.5) +
  labs(title = "Probability Distributions",

       x = "Response Categories / Counts",
       y = "Density / Probability") +
  scale_x_discrete(labels = c("Failure", "Success")) +
  theme_minimal() +
  theme(legend.position = "top") +
  scale_color_manual(values = c("blue", "red", "green"))
```

```
# Display summary statistics

cat("Summary Statistics for the Frequency Distribution:\n")
cat("Mean of Frequencies: ", mean(response_data$frequency), "\n")
cat("Standard Deviation of Frequencies: ", sd(response_data$frequency), "\n")
cat("Total Responses: ", sum(response_data$frequency), "\n")
cat("Probability of 'Strongly Disagree': ", p_bernoulli, "\n")
cat("Mean for Normal Distribution (mu): ", mu, "\n")
cat("Standard Deviation for Normal Distribution (sigma): ", sigma, "\n")
cat("Lambda for Poisson Distribution: ", lambda, "\n")
```

OUTPUT:

```
> # Display summary statistics
> cat("Summary Statistics for the Frequency Distribution:\n")
Summary Statistics for the Frequency Distribution:
> cat("Mean of Frequencies: ", mean(response_data$frequency), "\n")
Mean of Frequencies:  20
> cat("Standard Deviation of Frequencies: ", sd(response_data$frequency), "\n")
Standard Deviation of Frequencies:  10
> cat("Total Responses: ", sum(response_data$frequency), "\n")
Total Responses:  100
> cat("Probability of 'Strongly Disagree': ", p_bernoulli, "\n")
Probability of 'Strongly Disagree':  0.05
> cat("Mean for Normal Distribution (mu): ", mu, "\n")
Mean for Normal Distribution (mu):  3
> cat("Standard Deviation for Normal Distribution (sigma): ", sigma, "\n")
Standard Deviation for Normal Distribution (sigma):  1.581139
> cat("Lambda for Poisson Distribution: ", lambda, "\n")
Lambda for Poisson Distribution:  20
> 
```

Graph:



Probability Distributions
Density / Probability
0.75
0.50
0.25
0.00
Failure
Success
Response Categories / Counts

Experiment 07: Program to implement one sample, two sample and paired sample t-tests for a sample data and analyse the results.

```
# Set seed for reproducibility
set.seed(123)

# One-sample data
one_sample_data<- rnorm(30, mean = 50, sd = 10)

# Two-sample data
two_sample_data1 <- rnorm(30, mean = 55, sd = 10)
two_sample_data2 <- rnorm(30, mean = 50, sd = 10)

# Paired sample data
paired_data_before<- rnorm(30, mean = 60, sd = 10)
paired_data_after<- paired_data_before + rnorm(30, mean = -2, sd = 5)

# One-sample t-test
one_sample_test<- t.test(one_sample_data, mu = 50)
cat("One-Sample t-Test Results:\n")
print(one_sample_test)

# Two-sample t-test
two_sample_test<- t.test(two_sample_data1, two_sample_data2)
cat("\nTwo-Sample t-Test Results:\n")
print(two_sample_test)

# Paired t-test
paired_test<- t.test(paired_data_before, paired_data_after, paired = TRUE)
cat("\nPaired Sample t-Test Results:\n")
print(paired_test)
```

OUTPUT:

```
> # One-sample t-test
> one_sample_test<- t.test(one_sample_data, mu = 50)
> cat("One-Sample t-Test Results:\n")
One-Sample t-Test Results:
> print(one_sample_test)

        One Sample t-test

data:  one_sample_data
t = -0.26299, df = 29, p-value = 0.7944
alternative hypothesis: true mean is not equal to 50
95 percent confidence interval:
 45.86573 53.19219
sample estimates:
mean of x
 49.52896
```

```
> cat("\nTwo-Sample t-Test Results:\n")

Two-Sample t-Test Results:
> print(two_sample_test)

	Welch Two Sample t-test

data:  two_sample_data1 and two_sample_data2
t = 2.9703, df = 57.904, p-value = 0.004325
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  2.132245 10.946114
sample estimates:
mean of x mean of y 
 56.78338  50.24420 
```

```
Paired Sample t-Test Results:
> print(paired_test)

	Paired t-test

data:  paired_data_before and paired_data_after
t = 2.7832, df = 29, p-value = 0.009373
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 0.7736567 5.0621473
sample estimates:
mean difference 
       2.917902 
```

Experiment 08: Program to implement One-way and Two-way ANOVA tests and analyze the results

```
# Load necessary library
library(dplyr)

# One-Way ANOVA
set.seed(123)  # For reproducibility
treatment_A<- c(23, 25, 20, 22, 26)
treatment_B<- c(30, 29, 31, 32, 28)
treatment_C<- c(35, 36, 34, 33, 37)

data_one_way<- data.frame(
value = c(treatment_A, treatment_B, treatment_C),
treatment = factor(rep(c("A", "B", "C"), each = 5))
)

one_way_anova<- aov(value ~ treatment, data = data_one_way)
print(summary(one_way_anova))

# Two-Way ANOVA
treatment_A_male<- c(23, 25, 20, 22, 26)
treatment_A_female<- c(30, 28, 31, 29, 32)
treatment_B_male<- c(27, 29, 30, 26, 25)
treatment_B_female<- c(35, 36, 34, 33, 37)


data_two_way<- data.frame(
value = c(treatment_A_male, treatment_A_female, treatment_B_male,
treatment_B_female),
treatment = factor(rep(c("A", "B"), each = 10)),
gender = factor(rep(c("Male", "Female"), times = 10))
)

two_way_anova<- aov(value ~ treatment * gender, data = data_two_way)
print(summary(two_way_anova))
```

OUTPUT:

```
> print(summary(one_way_anova))
            Df Sum Sq Mean Sq F value   Pr(>F)    
treatment    2  350.8  175.40   49.18 1.65e-06 ***
Residuals   12   42.8    3.57                     
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
> # Two-Way ANOVA
> treatment_A_male<- c(23, 25, 20, 22, 26)
> treatment_A_female<- c(30, 28, 31, 29, 32)
> treatment_B_male<- c(27, 29, 30, 26, 25)
> treatment_B_female<- c(35, 36, 34, 33, 37)
>
> data_two_way<- data.frame(
+   value = c(treatment_A_male, treatment_A_female, treatment_B_mal
e, treatment_B_female),
+   treatment = factor(rep(c("A", "B"), each = 10)),
+   gender = factor(rep(c("Male", "Female"), times = 10))
+ )
>
> two_way_anova<- aov(value ~ treatment * gender, data = data_two_wa
y)
> print(summary(two_way_anova))
                 Df Sum Sq Mean Sq F value Pr(>F)  
treatment         1  105.8  105.80   5.829 0.0281 *
gender            1   28.8   28.80   1.587 0.2259  
treatment:gender  1    0.8    0.80   0.044 0.8364  
Residuals        16  290.4   18.15                 
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Experiment 09: Program to implement correlation, rank correlation and regression and plot x-y plot and heat maps of correlation matrices.

```
# Load necessary libraries
library(ggplot2)   # For plotting
library(dplyr)     # For data manipulation
library(reshape2)  # For reshaping data for heatmaps
library(corrplot)  # For correlation plot


# Load the mtcars dataset
data(mtcars)

# 1. Correlation
correlation_matrix<- cor(mtcars)
print("Correlation Matrix:")
print(correlation_matrix)


# 2. Rank Correlation (Spearman)
rank_correlation_matrix<- cor(mtcars, method = "spearman")
print("Rank Correlation Matrix (Spearman):")
print(rank_correlation_matrix)


# 3. Linear Regression Example
# Let's predict mpg based on wt and hp
model<- lm(mpg ~ wt + hp, data = mtcars)
summary(model)


# Predictions and residuals
mtcars$predicted_mpg<- predict(model)
mtcars$residuals<- residuals(model)


# 4. Plotting x-y plot
# Scatter plot of actual vs predicted mpg
ggplot(mtcars, aes(x = predicted_mpg, y = mpg)) +
geom_point(color = 'blue') +
geom_smooth(method = 'lm', color = 'red') +
labs(title = "Actual vs Predicted MPG",
     x = "Predicted MPG",
```

```
  y = "Actual MPG") +
theme_minimal()


# 5. Heatmap of Correlation Matrix
# Reshape the correlation matrix
cor_melted<- melt(correlation_matrix)


# Create the heatmap
ggplot(cor_melted, aes(Var1, Var2, fill = value)) +
geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
midpoint = 0, limit = c(-1, 1), space = "Lab",
name="Correlation") +
theme_minimal() +
labs(title = "Heatmap of Correlation Matrix") +
theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))


# 6. Correlation Plot
corrplot(correlation_matrix, method = "circle", type = "upper",
order = "hclust", tl.col = "black", tl.srt = 45,
title = "Correlation Plot")
```

OUTPUT:

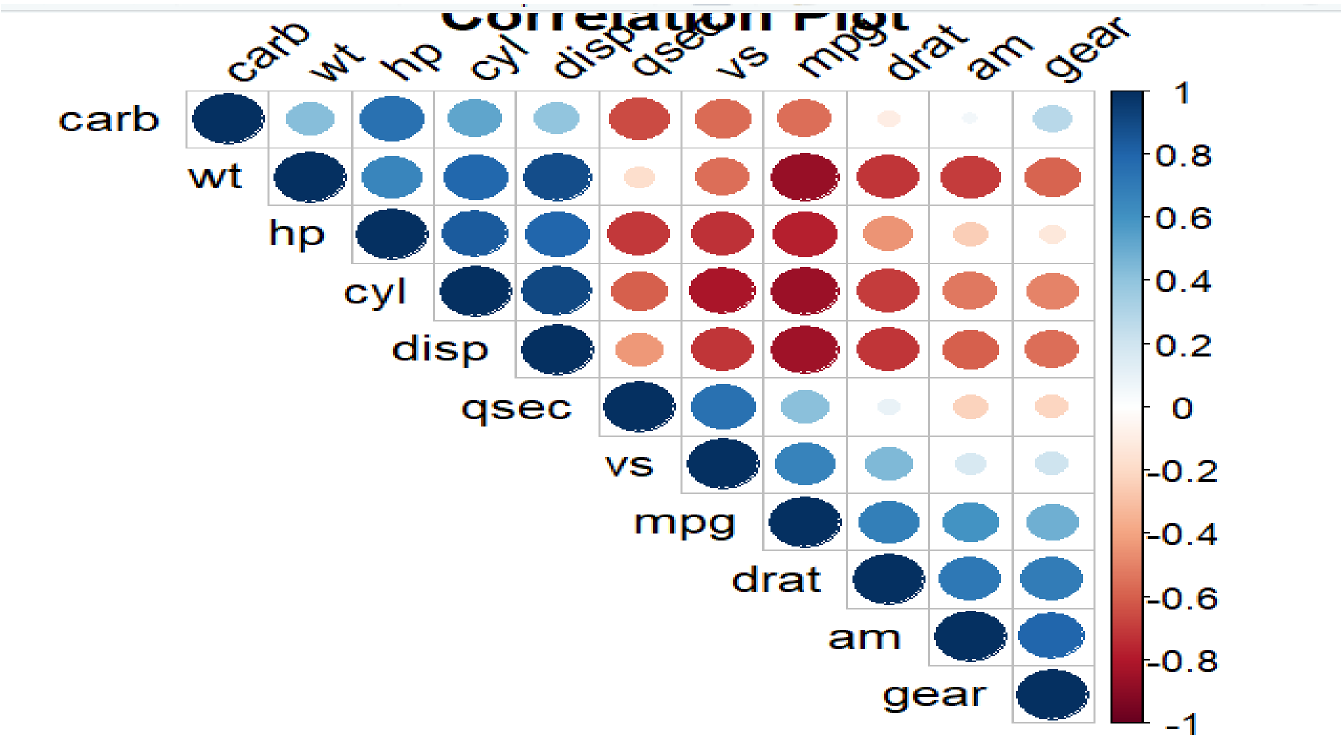
```
> print("Correlation Matrix:")
[1] "Correlation Matrix:"
> print(correlation_matrix)
            mpg        cyl       disp         hp        drat         wt        qsec         vs          am       gear
mpg   1.0000000 -0.8521620 -0.8475514 -0.7761684  0.68117191 -0.8676594  0.41868403  0.6640389  0.59983243  0.4802848
cyl  -0.8521620  1.0000000  0.9020329  0.8324475 -0.69993811  0.7824958 -0.59124207 -0.8108118 -0.52260705 -0.4926866
disp -0.8475514  0.9020329  1.0000000  0.7909486 -0.71021393  0.8879799 -0.43369788 -0.7104159 -0.59122704 -0.5555692
hp   -0.7761684  0.8324475  0.7909486  1.0000000 -0.44875912  0.6587479 -0.70822339 -0.7230967 -0.24320426 -0.1257043
drat  0.6811719 -0.6999381 -0.7102139 -0.4487591  1.00000000 -0.7124406  0.09120476  0.4402785  0.71271113  0.6996101
wt   -0.8676594  0.7824958  0.8879799  0.6587479 -0.71244065  1.0000000 -0.17471588 -0.5549157 -0.69249526 -0.5832870
qsec  0.4186840 -0.5912421 -0.4336979 -0.7082234  0.09120476 -0.1747159  1.00000000  0.7445354 -0.22986086 -0.2126822
vs    0.6640389 -0.8108118 -0.7104159 -0.7230967  0.44027846 -0.5549157  0.74453544  1.0000000  0.16834512  0.2060233
am    0.5998324 -0.5226070 -0.5912270 -0.2432043  0.71271113 -0.6924953 -0.22986086  0.1683451  1.00000000  0.7940588
gear  0.4802848 -0.4926866 -0.5555692 -0.1257043  0.69961013 -0.5832870 -0.21268223  0.2060233  0.79405876  1.0000000
carb -0.5509251  0.5269883  0.3949769  0.7498125 -0.09078980  0.4276059 -0.65624923 -0.5696071  0.05753435  0.2740728
            carb
mpg  -0.55092507
cyl   0.52698829
disp  0.39497686
hp    0.74981247
drat -0.09078980
wt    0.42760594
qsec -0.65624923
vs   -0.56960714
am    0.05753435
gear  0.27407284
carb  1.00000000
>
```

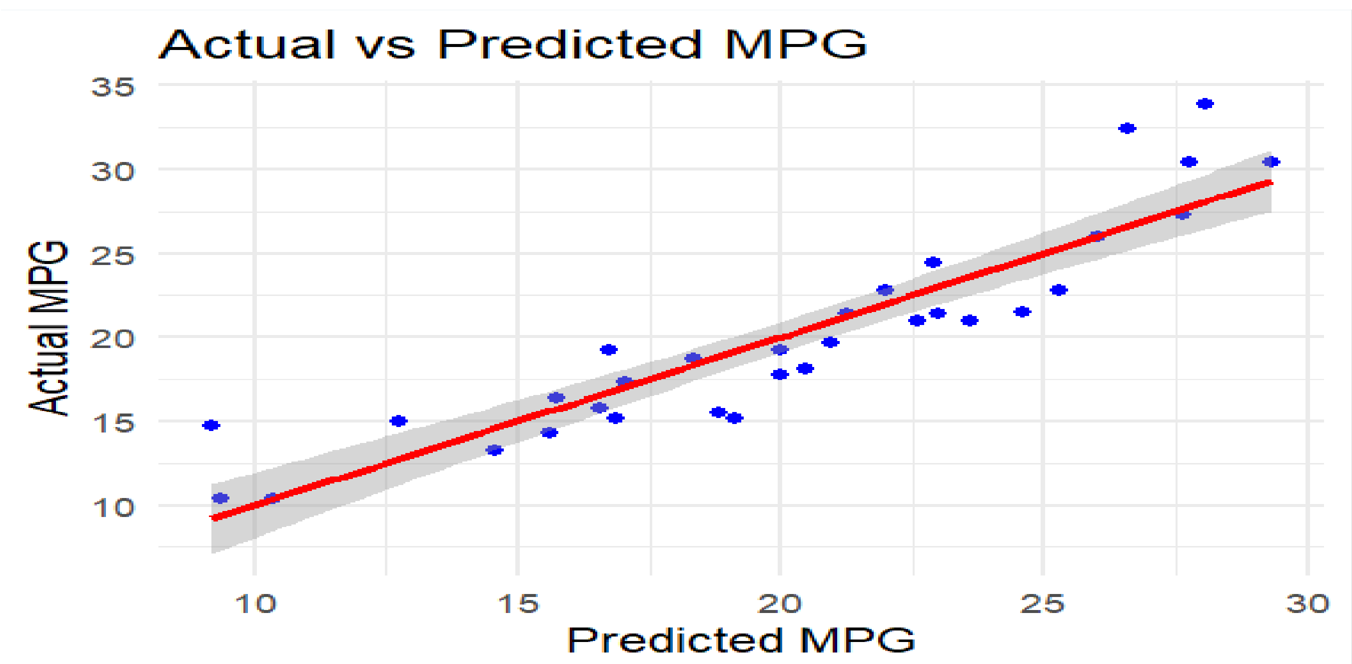
```
> print("Rank Correlation Matrix (Spearman):")
[1] "Rank Correlation Matrix (Spearman):"
> print(rank_correlation_matrix)
            mpg        cyl       disp         hp        drat         wt        qsec         vs          am       gear
mpg   1.0000000 -0.9108013 -0.9088824 -0.8946646  0.65145546 -0.8864220  0.46693575  0.7065968  0.56200569  0.5427816
cyl  -0.9108013  1.0000000  0.9276516  0.9017909 -0.67888119  0.8577282 -0.57235095 -0.8137890 -0.52207118 -0.5643105
disp -0.9088824  0.9276516  1.0000000  0.8510426 -0.68359210  0.8977064 -0.45978176 -0.7236643 -0.62406767 -0.5944703
hp   -0.8946646  0.9017909  0.8510426  1.0000000 -0.52012499  0.7746767 -0.66660602 -0.7515934 -0.36232756 -0.3314016
drat  0.6514555 -0.6788812 -0.6835921 -0.5201250  1.00000000 -0.7503904  0.09186863  0.4474575  0.68657079  0.7448162
wt   -0.8864220  0.8577282  0.8977064  0.7746767 -0.75039041  1.0000000 -0.22540120 -0.5870162 -0.73771259 -0.6761284
qsec  0.4669358 -0.5723509 -0.4597818 -0.6666060  0.09186863 -0.2254012  1.00000000  0.7915715 -0.20333211 -0.1481997
vs    0.7065968 -0.8137890 -0.7236643 -0.7515934  0.44745745 -0.5870162  0.79157148  1.0000000  0.16834512  0.2826617
am    0.5620057 -0.5220712 -0.6240677 -0.3623276  0.68657079 -0.7377126 -0.20333211  0.1683451  1.00000000  0.8076880
gear  0.5427816 -0.5643105 -0.5944703 -0.3314016  0.74481617 -0.6761284 -0.14819967  0.2826617  0.80768800  1.0000000
carb -0.6574976  0.5800680  0.5397781  0.7333794 -0.12522294  0.4998120 -0.65871814 -0.6336948 -0.06436525  0.1148870
            carb
mpg  -0.65749764
cyl   0.58006798
disp  0.53977806
hp    0.73337937
drat -0.12522294
wt    0.49981205
qsec -0.65871814
vs   -0.63369482
am   -0.06436525
gear  0.11488698
carb  1.00000000
>
```

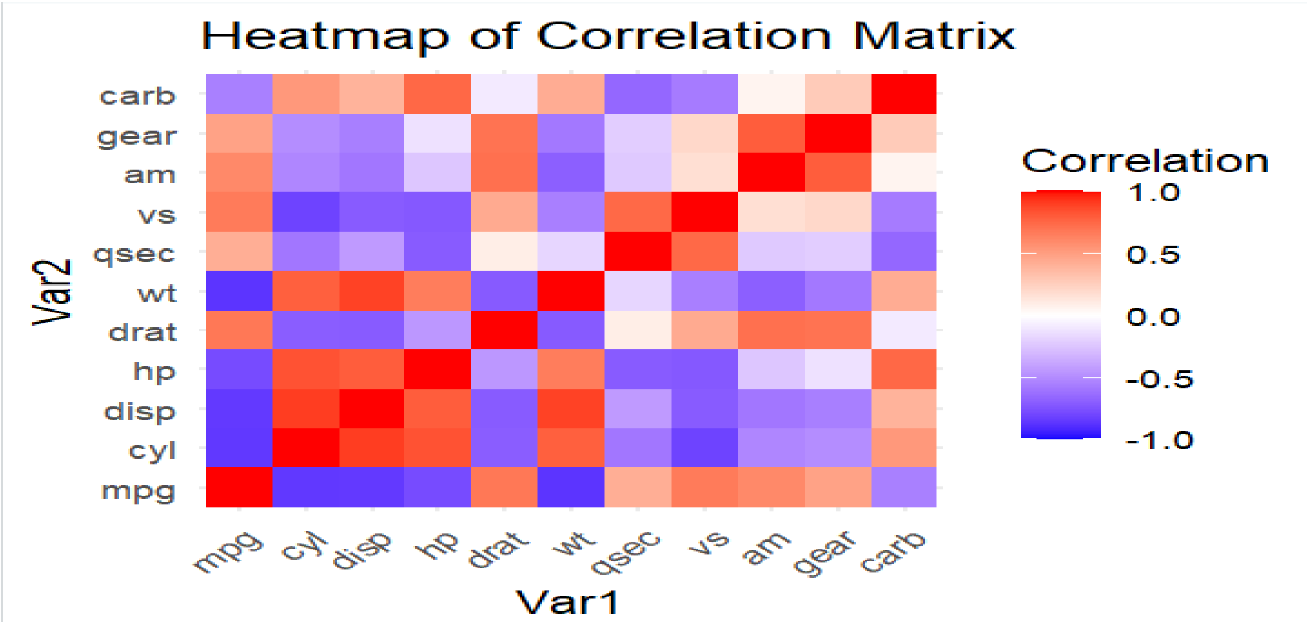
```
Call:
lm(formula = mpg ~ wt + hp, data = mtcars)

Residuals:
   Min     1Q Median     3Q    Max 
-3.941 -1.600 -0.182  1.050  5.854 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 37.22727    1.59879  23.285  < 2e-16 ***
wt          -3.87783    0.63273  -6.129 1.12e-06 ***
hp          -0.03177    0.00903  -3.519  0.00145 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.593 on 29 degrees of freedom
Multiple R-squared:  0.8268,	Adjusted R-squared:  0.8148 
F-statistic: 69.21 on 2 and 29 DF,  p-value: 9.109e-12

>
```

```
> # 4. Plotting x-y plot
> # Scatter plot of actual vs predicted mpg
> ggplot(mtcars, aes(x = predicted_mpg, y = mpg)) +
+   geom_point(color = 'blue') +
+   geom_smooth(method = 'lm', color = 'red') +
+   labs(title = "Actual vs Predicted MPG",
+        x = "Predicted MPG",
+        y = "Actual MPG") +
+   theme_minimal()
`geom_smooth()` using formula = 'y ~ x'
>
```

Graph:


Correlation Plot
carb
wt
hp
cyl
disp
qsec
vs
mpg
drat
am
gear
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1


Actual vs Predicted MPG
Actual MPG
Predicted MPG
35
30
25
20
15
10
10
15
20
25
30


Heatmap of Correlation Matrix
carb
gear
am
vs
qsec
wt
drat
hp
disp
cyl
mpg
Var2
mpg
cyl
disp
hp
drat
wt
qsec
vs
am
gear
carb
Var1
Correlation
1.0
0.5
0.0
-0.5
-1.0

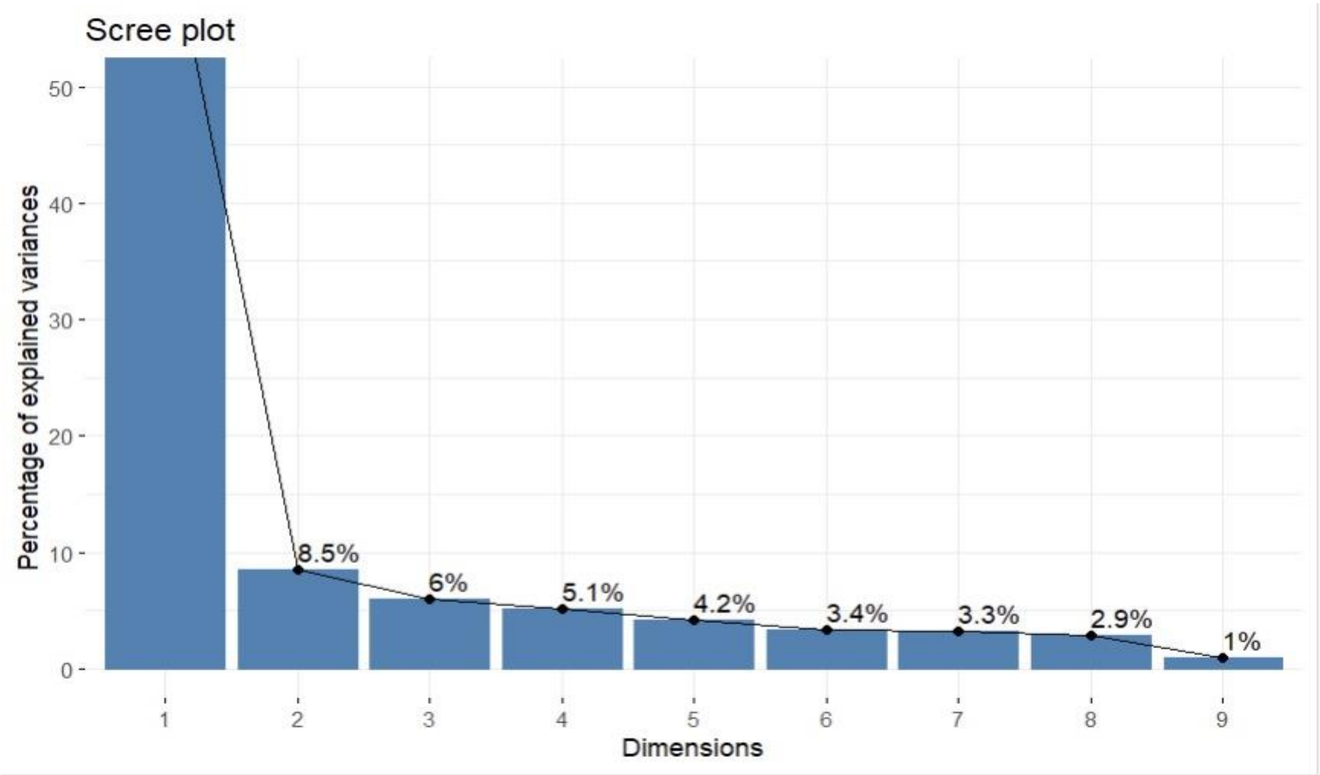
Experiment 10: Program to implement PCA for Wisconsin dataset, visualize and analyze the results.

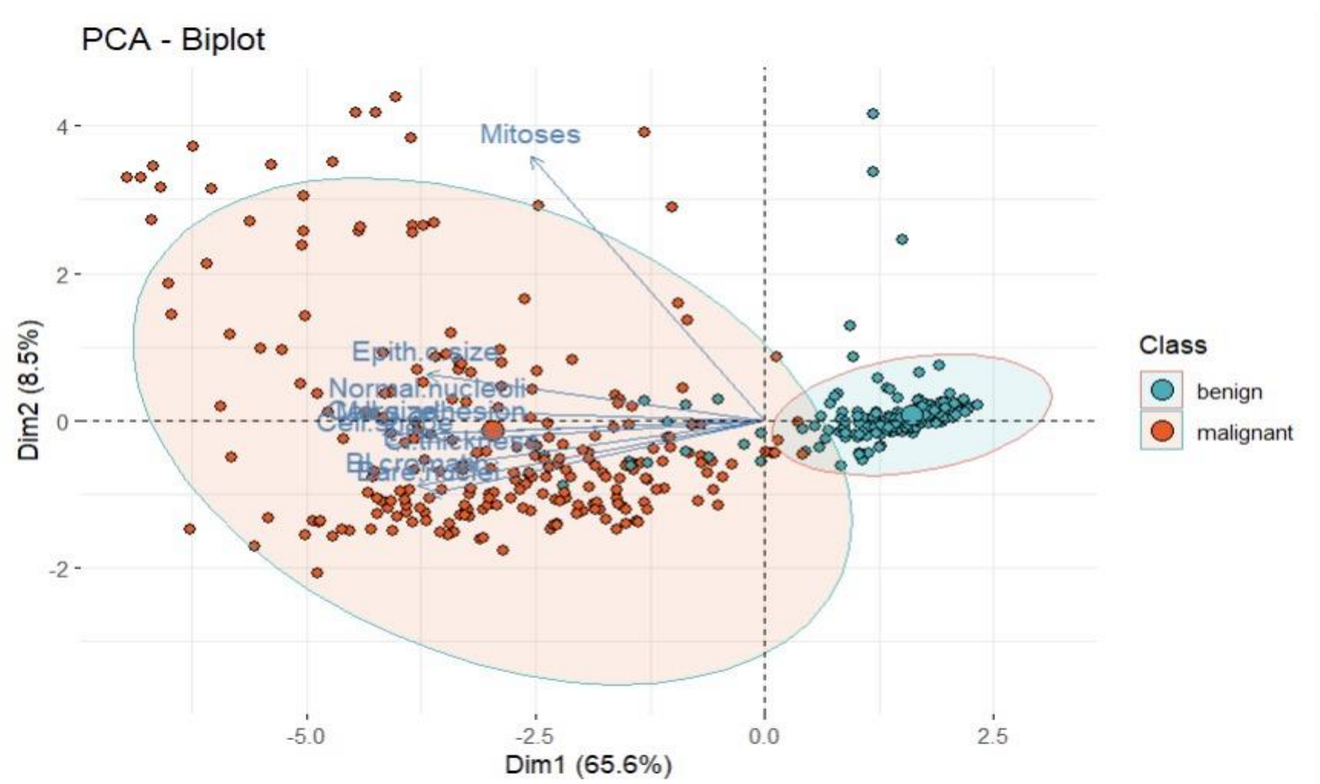
```
install.packages(c("mlbench", "ggplot2", "dplyr", "factoextra"))
library(mlbench)
library(ggplot2)
library(dplyr)
library(factoextra)

data("BreastCancer", package = "mlbench")
bc_data <- BreastCancer %>%
select(-Id) %>%
na.omit()

bc_data$Class <- as.factor(bc_data$Class)
numeric_data <- bc_data %>%
select(-Class) %>%
mutate(across(everything(), as.numeric))

scaled_data <- scale(numeric_data)

pca_result <- prcomp(scaled_data, center = TRUE, scale. = TRUE)

fviz_screeplot(pca_result, addlabels = TRUE, ylim = c(0, 50))

fviz_pca_biplot(pca_result,
geom.ind = "point",
pointshape = 21,
pointsize = 2,

fill.ind = as.factor(bc_data$Class),
palette = c("#00AFBB", "#FC4E07"),
addEllipses = TRUE,
legend.title = "Class")

summary(pca_result)
```

<u>Output:</u>

```
Importance of components:
                          PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9
Standard deviation     2.4302 0.87512 0.73417 0.67979 0.61688 0.55010 0.54274 0.51074 0.29730
Proportion of Variance 0.6562 0.08509 0.05989 0.05135 0.04228 0.03362 0.03273 0.02898 0.00982
Cumulative Proportion  0.6562 0.74132 0.80121 0.85256 0.89484 0.92847 0.96120 0.99018 1.00000
```

Graph:


Scree plot
Percentage of explained variances
Dimensions
50
40
30
20
10
0
1
2
3
4
5
6
7
8
9
8.5%
6%
5.1%
4.2%
3.4%
3.3%
2.9%
1%


PCA - Biplot
Mitoses
Epith.c.size
Normal.nucleoli
Cell.shape
Cl.thickness
Bl.cromatin
Bare.nuclei
Dim2 (8.5%)
Dim1 (65.6%)
Class
benign
malignant
4
2
0
-2
-5.0
-2.5
0.0
2.5

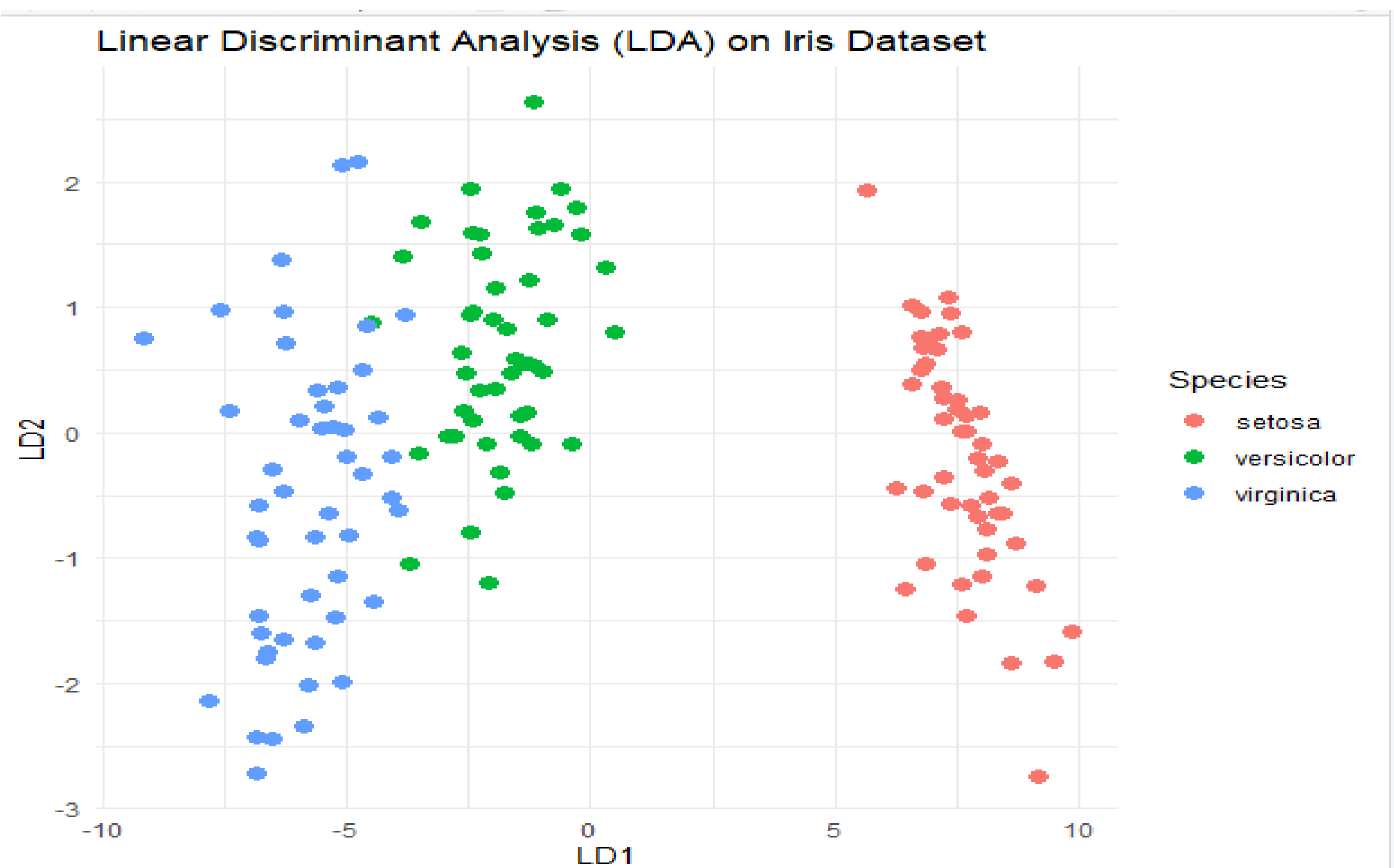
Experiment 11: Program to implement the working of linear discriminant analysis using iris dataset and visualize the results.

```
# Load necessary libraries
library(MASS)  # For LDA
library(ggplot2)  # For visualization

# Load the iris dataset
data(iris)

# Train the LDA model
lda_model <- lda(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data
= iris)

# Print the model summary
print(lda_model)

# Predict the class labels using the LDA model
lda_predictions <- predict(lda_model, iris)

# Add the predicted values to the original dataset
iris$lda_pred <- lda_predictions$class

# Visualize the results using ggplot2
# Plot the first two components of the LDA (LD1 and LD2)
lda_df <- data.frame(LD1 = lda_predictions$x[, 1], LD2 = lda_predictions$x[, 2], Species
= iris$Species)

ggplot(lda_df, aes(x = LD1, y = LD2, color = Species)) +
  geom_point(size = 3) +
  labs(title = "Linear Discriminant Analysis (LDA) on Iris Dataset", x = "LD1", y = "LD2")
+
  theme_minimal()
```

Output:

```
Call:
lda(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
    data = iris)

Prior probabilities of groups:
    setosa versicolor  virginica 
 0.3333333  0.3333333  0.3333333 

Group means:
           Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa            5.006       3.428        1.462       0.246
versicolor        5.936       2.770        4.260       1.326
virginica         6.588       2.974        5.552       2.026

Coefficients of linear discriminants:
                    LD1         LD2
Sepal.Length  0.8293776 -0.02410215
Sepal.Width   1.5344731 -2.16452123
Petal.Length -2.2012117  0.93192121
Petal.Width  -2.8104603 -2.83918785

Proportion of trace:
   LD1    LD2 
0.9912 0.0088 
>
```

Graph:



Experiment 12: Program to Implement multiple linear regression using iris dataset, visualize and analyze the results.

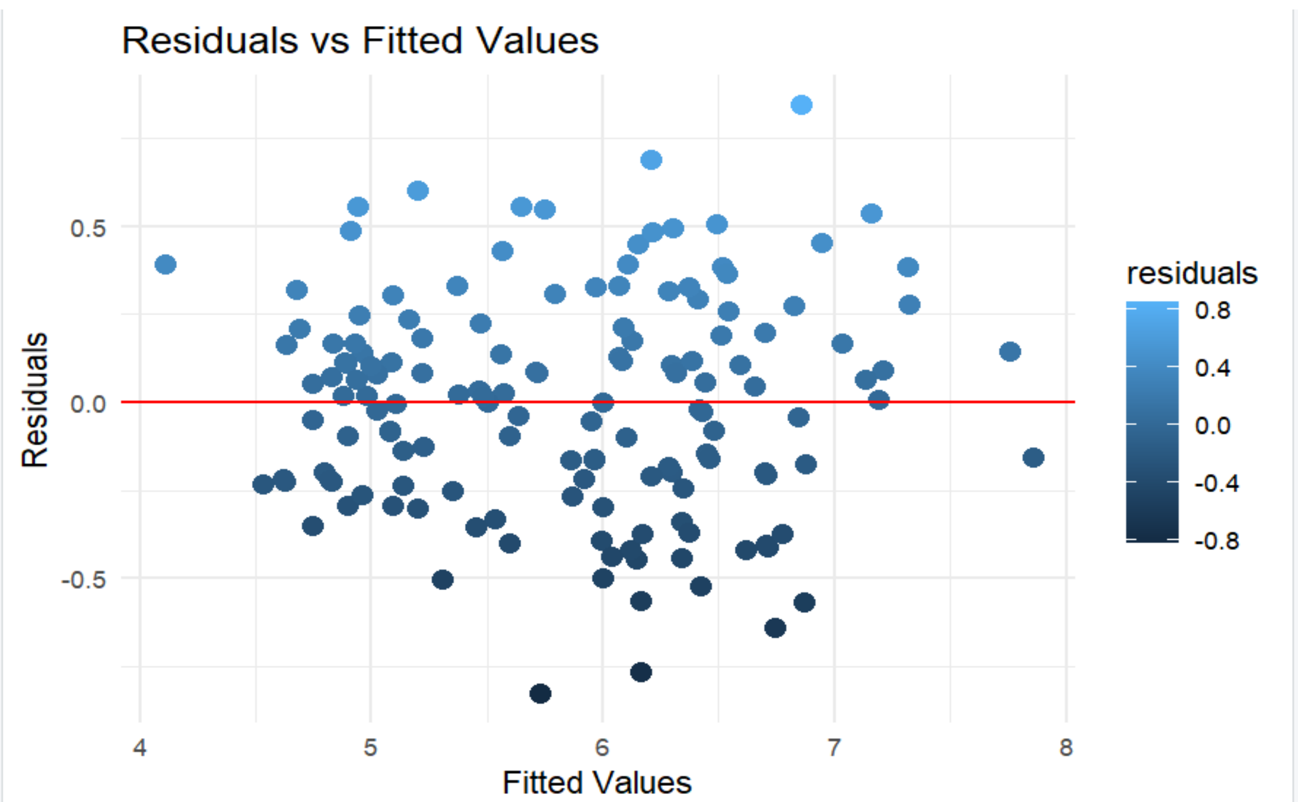
```
# Load necessary libraries
library(ggplot2)
library(caret)

# Load the iris dataset
data(iris)

# Display the first few rows of the iris dataset
head(iris)

# Fit the multiple linear regression model
# Predicting Sepal.Length based on Sepal.Width, Petal.Length, and Petal.Width
model <- lm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width, data = iris)

# Display the summary of the model to analyze coefficients and statistics
summary(model)

# Model diagnostic plots (checking residuals)
par(mfrow = c(2, 2))  # 2x2 grid for plots
plot(model)

# Visualize the relationship between the predicted and actual values
predicted_values <- predict(model, newdata = iris)

# Scatter plot of actual vs predicted values
ggplot(iris, aes(x = Sepal.Length, y = predicted_values)) +
  geom_point(aes(color = Species), size = 3) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +
  labs(x = "Actual Sepal Length", y = "Predicted Sepal Length", title = "Actual vs Predicted
Sepal Length") +
  theme_minimal()

# Checking residuals vs fitted values
ggplot(data.frame(fitted = fitted(model), residuals = residuals(model)), aes(x = fitted, y =
residuals)) +
  geom_point(aes(color = residuals), size = 3) +
  geom_hline(yintercept = 0, color = "red") +
  labs(x = "Fitted Values", y = "Residuals", title = "Residuals vs Fitted Values") +
  theme_minimal()

# Evaluate model accuracy using RMSE (Root Mean Squared Error)
rmse <- sqrt(mean(residuals(model)^2))
cat("Root Mean Squared Error (RMSE):", rmse, "\n")
```

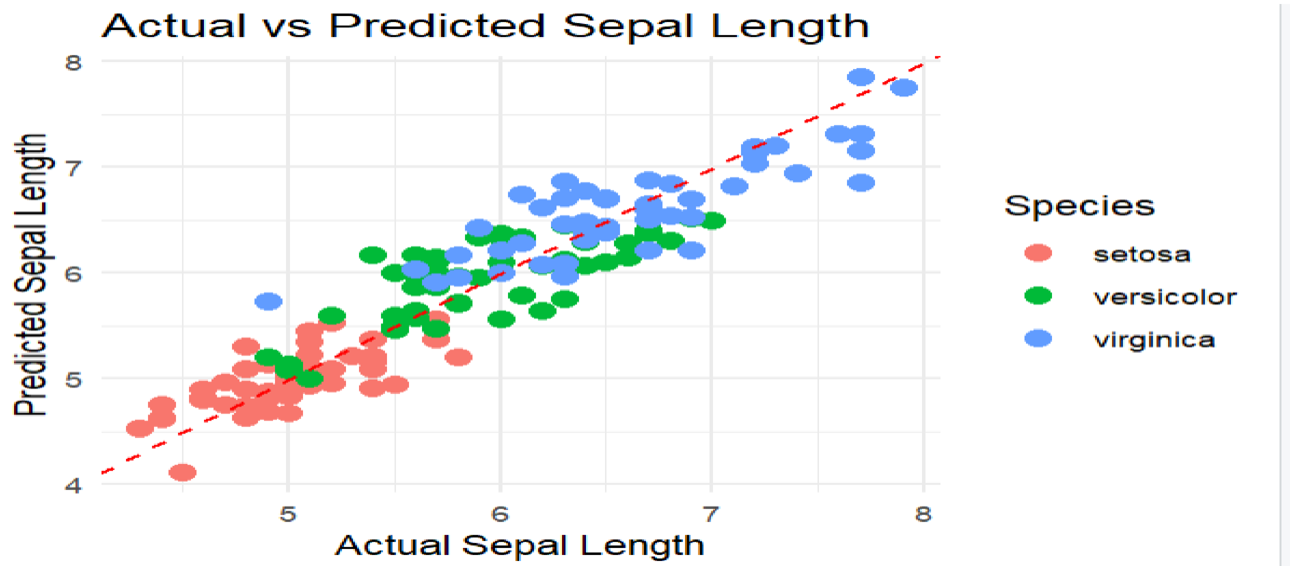
# Analyzing Variance Inflation Factor (VIF) for multicollinearity check
library(car)
vif(model)

Output:

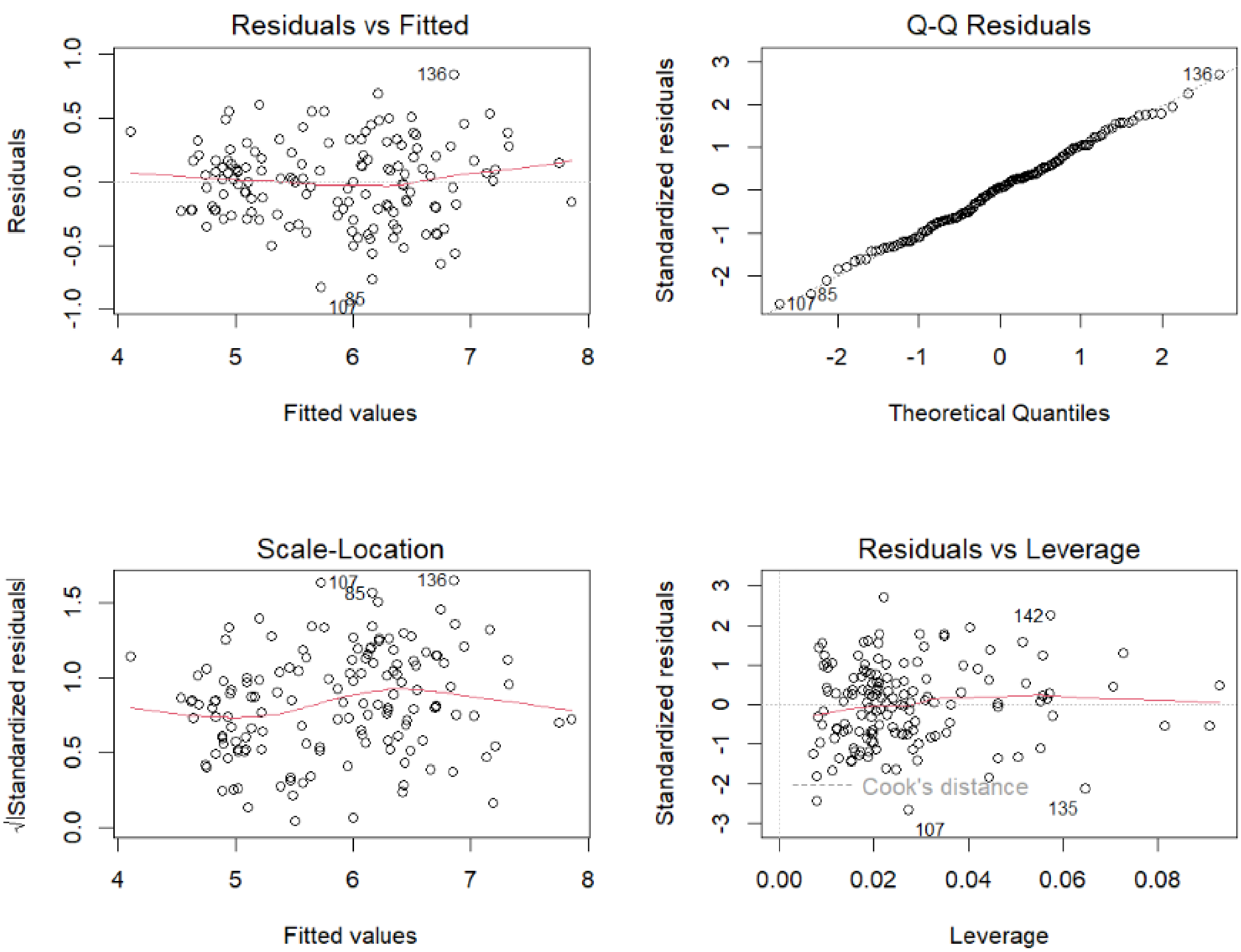
```
> cat("Root Mean Squared Error (RMSE):", rmse, "\n")
Root Mean Squared Error (RMSE): 0.3103268
>
> # Analyzing Variance Inflation Factor (VIF) for multicollinearity check
> library(car)
> vif(model)
 Sepal.Width Petal.Length  Petal.Width
    1.270815    15.097572    14.234335
```

Graph:





Actual vs Predicted Sepal Length
Predicted Sepal Length
Actual Sepal Length
Species
setosa
versicolor
virginica



Residuals vs Fitted
Residuals
Fitted values
Q-Q Residuals
Standardized residuals
Theoretical Quantiles
Scale-Location
√|Standardized residuals|
Fitted values
Residuals vs Leverage
Standardized residuals
Cook's distance
Leverage