FORM NO.-F/TL/021

REV.00Date 20.03.2020



**RECORD NOTEBOOK**

**STATISTICAL INFERRENCE LAB-(EBDS22L01) 2024-2025(ODD SEMESTER)**

**DEPARTMENT**

**Of**

**COMPUTER SCIENCE AND ENGINEERING**

**NAME : R. JAYADITHYA**

**REGISTER NO : 221191101064**

**COURSE : B.TECH-CSE(DS&AI)**

**YEAR/SEM/SEC : III/V/B**

FORM NO.-F/TL/021

REV.00Date 20.03.2020



**BONAFIDE CERTIFICATE**

Register No : **221191101064**

Name of Lab : **STATISTICAL INFERRENCE LAB-(EBDS22L01)**

Department : **COMPUTER SCIENCE AND ENGINEERING ( DS& AI)**

Certified that this is the bonafide record of work done by **R. JAYADITHYA, 221191101064**

of III Year B.Tech(CSE-DS& AI), Sec-B in the “**STATISTICAL INFERRENCE LAB-**

**(EBDS22L01)**” during the year 2024-2025.

**Signature of Lab-in-Charge Signature of Head of Dept**

Submitted for the Practical Examination held on **-------------------------**

**Internal Examiner External Examiner**

## CONTENTS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ex. No** | **Date** | **Name of the Experiment** | **Page No** | **Marks** | **Signature of the Staff** |
| **1** |  | For a given of Number find Mean, Median, Mode. | 1 |  |  |
| **2** |  | Calculate Quartile and Standard Deviation to a population. | 4 |  |  |
| **3** |  | Write a program for Probability Distribution. | 7 |  |  |
| **4** |  | Write a program to implement Bayes’ Theorem. | 10 |  |  |
| **5** |  | Estimate the Variability of data using programming language. | 13 |  |  |
| **6** |  | Calculate the Correlation of a population. | 16 |  |  |
| **7** |  | Find the Normal and Binomial Distribution of a sample data set | 19 |  |  |
| **8** |  | Perform a Hypothesis testing in a data set. | 24 |  |  |
| **9** | . | Calculate Simple and Multiple Linear Regressions. | 28 |  |  |
| **10** |  | Perform a Chi-Square Test of a sample data. | 32 |  |  |

|  |  |  |
| --- | --- | --- |
| **EXP NO** | **FOR A GIVEN OF NUMBER FIND MEAN, MEDIAN, MODE** | **DATE** |
| **1** |  |

### Aim:

To find Mean, Median, Mode for a given number.

### Algorithm:

1. **Load Necessary Library**:
   * Check if the dplyr package is installed. If not, install and load it.

### Define Mode Function:

* + Create get\_mode(v) that finds unique values, counts frequencies, and returns the most frequent value.

### Create Data Vector:

* + Define a numeric vector data\_vector with sample values.

### Calculate Mean, Median, and Mode:

* + Use mean(), median(), and the custom get\_mode() function to calculate each statistic.

### Print Results:

* + Display the calculated mean, median, and mode using cat().

### Program:

# Load necessary library if (!require("dplyr")) { install.packages("dplyr")

}

library(dplyr)

# Custom function to calculate mode get\_mode <- function(v) {

uniq\_v <- unique(v) uniq\_v[which.max(tabulate(match(v, uniq\_v)))]

}

# Sample numeric vector

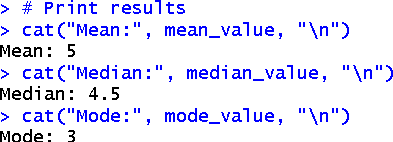
data\_vector <- c(2, 3, 5, 3, 8, 3, 4, 5, 8, 9)

# Calculate mean, median, and mode mean\_value <- mean(data\_vector) median\_value <- median(data\_vector) mode\_value <- get\_mode(data\_vector)

# Print results

cat("Mean:", mean\_value, "\n") cat("Median:", median\_value, "\n") cat("Mode:", mode\_value, "\n")

### Output:



**Result:**

Thus, the program was executed successfully and the output has been verified.

|  |  |  |
| --- | --- | --- |
| **EXP**  **NO** | **CALCULATE QUARTILE AND STANDARD DEVIATION TO A**  **POPULATION** | **DATE** |
| **2** |  |

### Aim:

To calculate Quartile and Standard Deviation to a population.

### Algorithm:

1. Create Data Vector: Define the data points in a vector.
2. Calculate Quartiles: Use the quantile() function to compute the quartiles.
3. Calculate Standard Deviation: Use the sd() function to compute the standard deviation.
4. Print Results: Output the quartiles and the standard deviation using cat() and print().

### Program:

# Sample numeric vector

data\_vector <- c(12, 15, 14, 10, 18, 20, 25, 30, 22, 28)

# Calculate quartiles

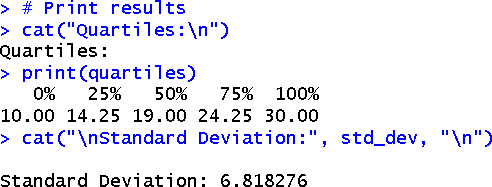
quartiles <- quantile(data\_vector)

# Calculate standard deviation std\_dev <- sd(data\_vector)

# Print results cat("Quartiles:\n") print(quartiles)

cat("\nStandard Deviation:", std\_dev, "\n")

### Output:



**Result:**

Thus, the program was executed successfully and the output has been verified.

## EXP NO

**3**

# WRITE A PROGRAM FOR PROBABILITY DISTRIBUTION

## DATE

### Aim:

To write a program for Probability Distribution.

### Algorithm:

1. Load the Required Library: Check if the ggplot2 package is installed. If not, install it, and then load the library using library().
2. Set Normal Distribution Parameters: Define the mean and standard deviation for the normal distribution.
3. Generate X Values: Create a sequence of x values ranging from -4 to 4 with a step of 0.1 using seq().
4. Calculate Probabilities: Compute the probability density function values for the normal distribution using dnorm() for the generated x values.
5. Create Data Frame for Plotting: Create a data frame with x values and their corresponding probability densities.
6. Plot the Normal Distribution: Use ggplot() to create a line plot of the normal distribution and fill the area under the curve with a semi-transparent color.
7. Display Plot Titles and Labels: Set the plot title, x-axis label, y-axis label, and apply a minimal theme.
8. Print Probability Values for Specific X: For x values of -2, 0, and 2, calculate and print the corresponding probability densities using dnorm() and cat().

### Program:

# Load necessary library if (!require("ggplot2")) {

install.packages("ggplot2")

}

library(ggplot2)

# Normal Distribution Parameters mean\_normal <- 0 # Mean

sd\_normal <- 1 # Standard Deviation

x\_values\_normal <- seq(-4, 4, by = 0.1) # Range of x values

# Calculate probabilities for Normal Distribution

normal\_probabilities <- dnorm(x\_values\_normal, mean = mean\_normal, sd = sd\_normal)

# Create a data frame for plotting

normal\_df <- data.frame(x = x\_values\_normal, y = normal\_probabilities)

# Plot Normal Distribution ggplot(normal\_df, aes(x, y)) + geom\_line(color = "blue") +

geom\_area(aes(y = y), fill = "lightblue", alpha = 0.5) + # Fill area under the curve ggtitle("Normal Distribution (mean = 0, sd = 1)") +

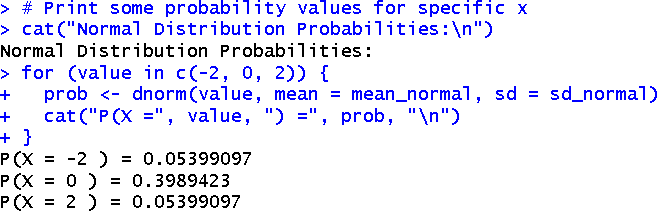
xlab("Value") + ylab("Density") + theme\_minimal()

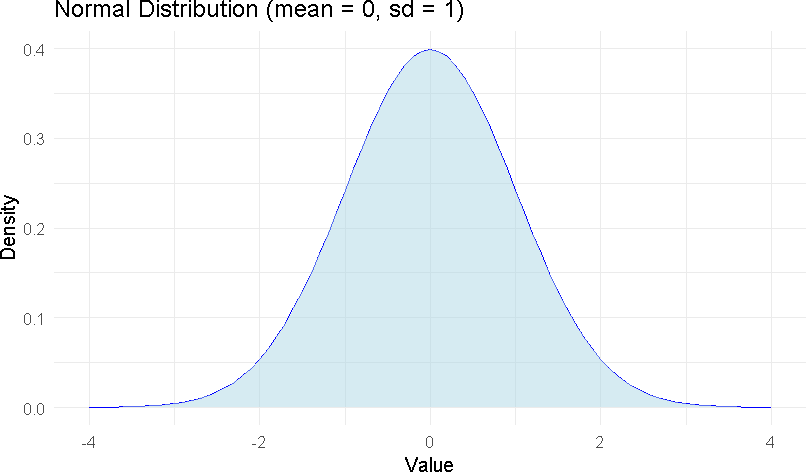
# Print some probability values for specific x cat("Normal Distribution Probabilities:\n") for (value in c(-2, 0, 2)) {

prob <- dnorm(value, mean = mean\_normal, sd = sd\_normal) cat("P(X =", value, ") =", prob, "\n")

}

### Output:





**Result:**

Thus, the program was executed successfully and the output has been verified.

## EXP NO

**4**

# WRITE A PROGRAM TO

**IMPLEMENT BAYES’ THEOREM**

## DATE

### Aim:

To write a program to implement Bayes’ Theorem.

### Algorithm:

1. Define Bayes' Theorem Function: Create a function bayes\_theorem() that takes prior and likelihood values as input and calculates the posterior probability using Bayes' Theorem formula.
2. Set Example Parameters:
   * Define prior\_A as the prior probability of event A.
   * Define prior\_B as the prior probability of event B.
   * Define likelihood\_A\_given\_B as the likelihood of A given B.
3. Calculate likelihood\_B using the total probability formula for event B.
4. Calculate Posterior Probability: Call the bayes\_theorem() function to compute the posterior probability P(A | B) using the given parameters.
5. Print Results: Output the prior probabilities P(A) and P(B) and the posterior probability P(A | B) using cat().

### Program:

# Function to calculate Bayes' Theorem

bayes\_theorem <- function(prior\_A, prior\_B, likelihood\_A\_given\_B, likelihood\_B) { # Calculate P(B | A) using Bayes' Theorem

posterior\_A\_given\_B <- (likelihood\_A\_given\_B \* prior\_A) / likelihood\_B return(posterior\_A\_given\_B)

}

# Example Parameters

prior\_A <- 0.1 # P(A): Prior probability of event A

prior\_B <- 0.2 # P(B): Prior probability of event B likelihood\_A\_given\_B <- 0.9 # P(A | B): Likelihood of A given B occurs

likelihood\_B <- (likelihood\_A\_given\_B \* prior\_A) + (0.5 \* (1 - prior\_A)) # Total probability of B

# Calculate P(A | B) using Bayes' Theorem

posterior\_A\_given\_B <- bayes\_theorem(prior\_A, prior\_B, likelihood\_A\_given\_B, likelihood\_B)

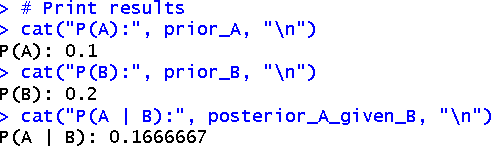
# Print results

cat("P(A):", prior\_A, "\n")

cat("P(B):", prior\_B, "\n")

cat("P(A | B):", posterior\_A\_given\_B, "\n")

### Output:



**Result:**

Thus, the program was executed successfully and the output has been verified.

|  |  |  |
| --- | --- | --- |
| **EXP**  **NO** | **ESTIMATE THE VARIABILITY OF DATA USING PROGRAMMING**  **LANGUAGE** | **DATE** |
| **5** |  |

### Aim:

To Estimate the Variability of data using programming language.

### Algorithm:

1. Define Sample Data: Create a numeric vector data\_vector containing the data points for analysis.
2. Calculate Variance: Use the var() function to compute the variance of the data in data\_vector.
3. Calculate Standard Deviation: Use the sd() function to compute the standard deviation of the data in data\_vector.
4. Calculate Interquartile Range (IQR): Use the IQR() function to compute the interquartile range of the data in data\_vector.
5. Print Results: Use the cat() function to output the calculated variance, standard deviation, and interquartile range (IQR).

### Program:

# Sample Data

data\_vector <- c(25, 30, 35, 40, 28, 32, 31, 29, 33, 34)

# Calculate Variability Measures # 1. Variance

variance <- var(data\_vector)

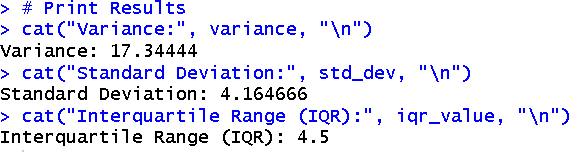
# 2. Standard Deviation std\_dev <- sd(data\_vector)

# 3. Interquartile Range (IQR) iqr\_value <- IQR(data\_vector)

# Print Results cat("Variance:", variance, "\n")

cat("Standard Deviation:", std\_dev, "\n") cat("Interquartile Range (IQR):", iqr\_value, "\n")

### Output:



**Result:**

Thus, the program was executed successfully and the output has been verified.

|  |  |  |
| --- | --- | --- |
| **EXP NO** | **CALCULATE THE CORRELATION OF A POPULATION** | **DATE** |
| **6** |  |

### Aim:

To Calculate the Correlation of a population

### Algorithm:

1. Set Seed for Reproducibility:
   * Set a random seed (set.seed(42)) to ensure consistent results in random data generation.
2. Generate Dataset:
   * Create a data frame with 100 random values for Height and Weight:
     + Height: Normally distributed with a mean of 170 cm and standard deviation of 10 cm.
     + Weight: Normally distributed with a mean of 65 kg and standard deviation of 10 kg.
3. Introduce Correlation:
   * Adjust the Weight variable by adding 0.7 kg for every 1 cm deviation from the average height of 170 cm to introduce a realistic correlation between height and weight.
4. Calculate Correlation:
   * Use the cor() function to compute the correlation coefficient between the Height and Weight variables.
5. Display the Result:
   * Print the correlation coefficient using cat().

### Program:

set.seed(42)

data <- data.frame(

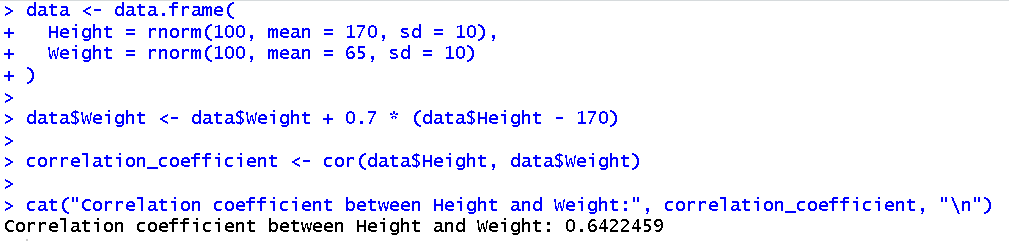
Height = rnorm(100, mean = 170, sd = 10), Weight = rnorm(100, mean = 65, sd = 10)

)

data$Weight <- data$Weight + 0.7 \* (data$Height - 170) correlation\_coefficient <- cor(data$Height, data$Weight)

cat("Correlation coefficient between Height and Weight:", correlation\_coefficient, "\n")

### Output :



**Result:**

Thus, the program was executed successfully and the output has been verified**.**

|  |  |  |
| --- | --- | --- |
| **EXP**  **NO** | **FIND THE NORMAL AND BINOMIAL DISTRIBUTION OF A SAMPLE DATA**  **SET** | **DATE** |
| **7** |  |

### Aim:

To find the Normal And Binomial Distribution of a sample data set.

### Algorithm:

**For Normal Distribution:**

1. Define the mean, standard deviation, and sample size for the normal distribution.
2. Generate random samples based on the defined normal distribution.
3. Calculate cumulative probabilities for specified values (e.g., 45 and 55).
4. Print the calculated probabilities.
5. Visualize the distribution of the generated samples using a density plot.

### Binomial Distribution:

1. Define the number of trials, probability of success, and sample size for the binomial distribution.
2. Generate random samples based on the defined binomial distribution.
3. Calculate the cumulative probability for a specific number of successes (e.g., 5).
4. Print the calculated probability.
5. Visualize the distribution of successes using a bar plot.

### Program:

# Load necessary libraries

# If you haven't installed the ggplot2 library yet, uncomment the next line # install.packages("ggplot2")

library(ggplot2)

# Normal Distribution Calculations

# Set parameters for the normal distribution mean\_normal <- 50

sd\_normal <- 10

n\_normal <- 1000 # Number of random samples

# Generate random samples from a normal distribution

normal\_samples <- rnorm(n\_normal, mean = mean\_normal, sd = sd\_normal)

# Calculate probabilities

prob\_normal\_45 <- pnorm(45, mean = mean\_normal, sd = sd\_normal) # P(X <= 45) prob\_normal\_55 <- pnorm(55, mean = mean\_normal, sd = sd\_normal) # P(X <= 55)

# Print probabilities

cat("Probability P(X <= 45):", prob\_normal\_45, "\n") cat("Probability P(X <= 55):", prob\_normal\_55, "\n")

# Plot the normal distribution

ggplot(data.frame(x = normal\_samples), aes(x)) + geom\_density(fill = "lightblue", alpha = 0.5) + labs(title = "Normal Distribution (mean = 50, sd = 10)",

x = "Value",

y = "Density") + theme\_minimal()

# Binomial Distribution Calculations

# Set parameters for the binomial distribution size\_binomial <- 10 # Number of trials

prob\_success <- 0.5 # Probability of success (e.g., coin flip) n\_binomial <- 1000 # Number of random samples

# Generate random samples from a binomial distribution

binomial\_samples <- rbinom(n\_binomial, size = size\_binomial, prob = prob\_success)

# Calculate probabilities

prob\_binomial\_5 <- pbinom(5, size = size\_binomial, prob = prob\_success) # P(X <= 5)

# Print probabilities

cat("Probability P(X <= 5):", prob\_binomial\_5, "\n")

# Plot the binomial distribution

binom\_df <- data.frame(successes = 0:size\_binomial,

probability = dbinom(0:size\_binomial, size = size\_binomial, prob = prob\_success))

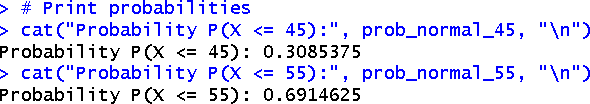
ggplot(binom\_df, aes(x = successes, y = probability)) + geom\_bar(stat = "identity", fill = "lightgreen") + labs(title = "Binomial Distribution (n = 10, p = 0.5)",

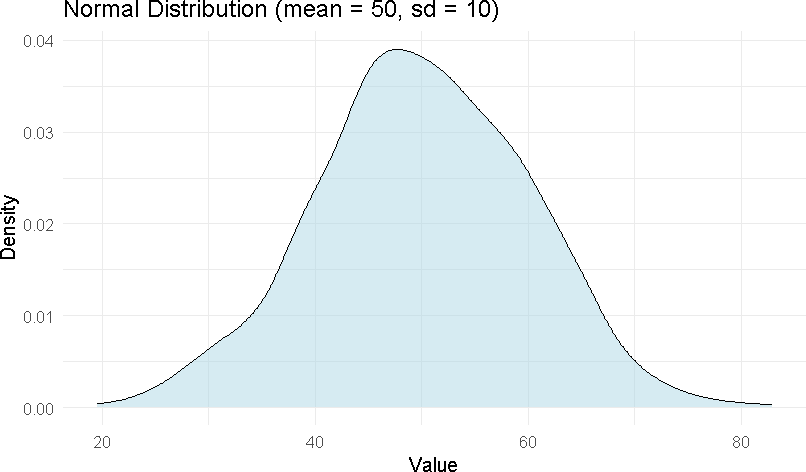
x = "Number of Successes", y = "Probability") +

theme\_minimal()

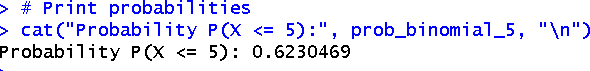
### Output :

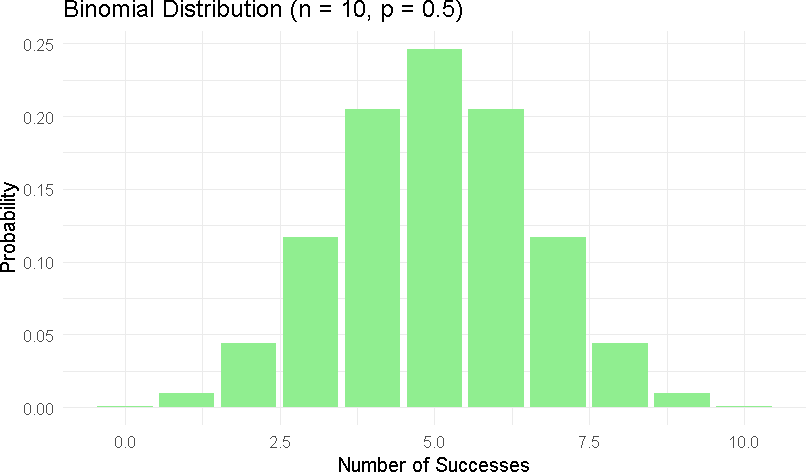
Normal Distribution:





Binomial Distribution:





### Result:

Thus, the program was executed successfully and the output has been verified.

|  |  |  |
| --- | --- | --- |
| **EXP NO** | **PERFORM A HYPOTHESIS TESTING IN A DATA SET** | **DATE** |
| **8** |  |

### Aim:

To perform a Hypothesis testing in a data set.

### Algorithm:

1. Set a seed for reproducibility using set.seed().
2. Define the sample size and population mean, then generate a sample dataset with a different mean using rnorm().
3. Calculate the sample mean with mean() and the sample standard deviation using sd().
4. Set the known population mean.
5. Perform a one-sample t-test with t.test() to compare the sample mean against the population mean.
6. Print the t-statistic, p-value, confidence interval, and sample mean from the t-test results.
7. State the null hypothesis (H0) and alternative hypothesis (H1).
8. Compare the p-value to the significance level (e.g., 0.05) to decide whether to reject or fail to reject the null hypothesis, and print the conclusion about the significance of the difference between the sample mean and the population mean.

### Program:

# Load necessary library

# If you haven't installed the dplyr library yet, uncomment the next line # install.packages("dplyr")

library(dplyr)

# Set seed for reproducibility set.seed(123)

# Generate a sample dataset sample\_size <- 30

population\_mean <- 100

sample\_data <- rnorm(sample\_size, mean = 102, sd = 15) # Sample with mean different from population mean

# Calculate sample mean and standard deviation sample\_mean <- mean(sample\_data)

sample\_sd <- sd(sample\_data)

# Print sample statistics

cat("Sample Mean:", sample\_mean, "\n") cat("Sample Standard Deviation:", sample\_sd, "\n")

# Set the known population mean mu <- population\_mean

# Perform a one-sample t-test

t\_test\_result <- t.test(sample\_data, mu = mu)

# Print t-test results

cat("t-statistic:", t\_test\_result$statistic, "\n") cat("p-value:", t\_test\_result$p.value, "\n")

cat("Confidence Interval:", t\_test\_result$conf.int, "\n")

cat("Sample Mean:", t\_test\_result$estimate, "\n")

cat("Null Hypothesis (H0): Sample mean is equal to population mean (", mu, ")\n", sep = "") cat("Alternative Hypothesis (H1): Sample mean is not equal to population mean (", mu, ")\n", sep = "")

# Interpretation

if (t\_test\_result$p.value < 0.05) {

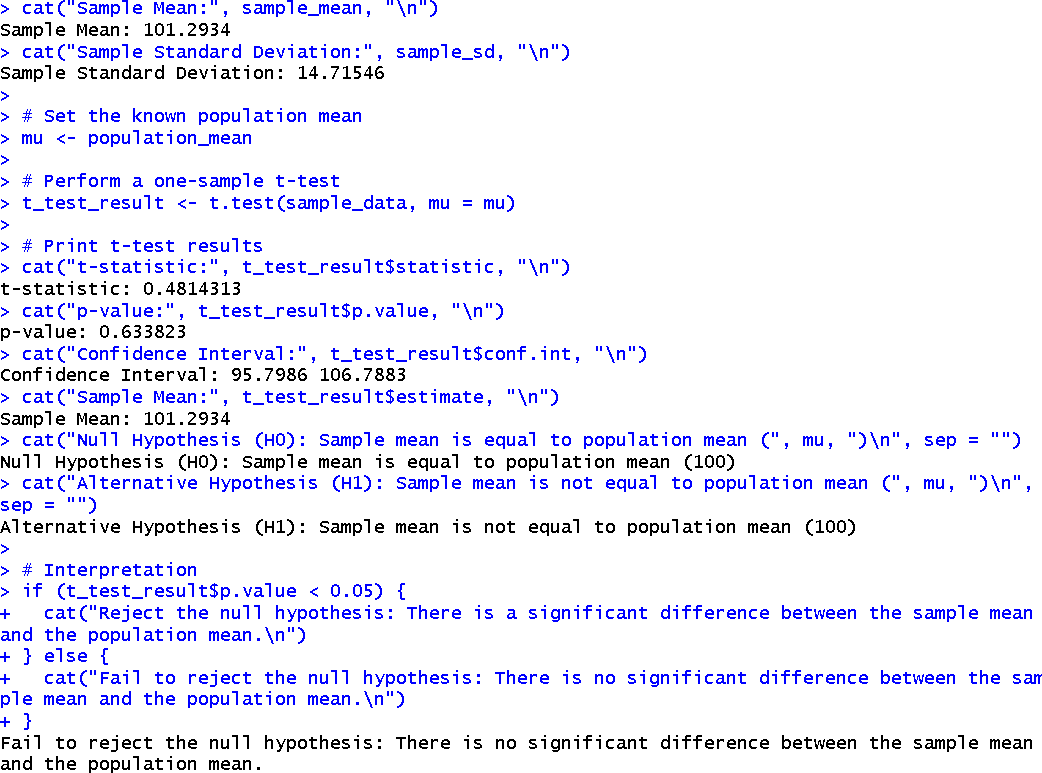
cat("Reject the null hypothesis: There is a significant difference between the sample mean and the population mean.\n")

} else {

cat("Fail to reject the null hypothesis: There is no significant difference between the sample mean and the population mean.\n")

}

### Output:



**Result:**

Thus, the program was executed successfully and the output has been verified.

|  |  |  |
| --- | --- | --- |
| **EXP**  **NO** | **CALCULATE SIMPLE AND MULTIPLE LINEAR REGRESSIONS** | **DATE** |
| **9** |  |

### Aim:

To calculate Simple and Multiple Linear Regressions.

### Algorithm:

1. Load Necessary Libraries: Load the ggplot2 library for creating visualizations.
2. Load the Dataset: Load the mtcars dataset into the R environment.
3. Simple Linear Regression: Fit a simple linear regression model using lm() to predict mpg based on wt.
4. Plot Simple Linear Regression: Create a scatter plot of wt versus mpg and add a linear regression line.
5. Multiple Linear Regression: Fit a multiple linear regression model using lm() to predict mpg based on wt, hp, and drat.
6. Make Predictions: Generate predicted values for mpg using the predict() function and add them to the dataset.
7. View Updated Dataset: Display the first few rows of the updated dataset with actual and predicted values.
8. Plot Actual vs. Predicted Values: Create a scatter plot of actual mpg versus predicted mpg, including a 45-degree reference line.

**Program:**

# Load necessary library library(ggplot2)

# Load the mtcars dataset data(mtcars)

# --- Simple Linear Regression ---

simple\_model <- lm(mpg ~ wt, data = mtcars) # Fit the model summary(simple\_model) # Summary of the model

# Plot the results for simple regression ggplot(mtcars, aes(x = wt, y = mpg)) + geom\_point() +

geom\_smooth(method = "lm", se = FALSE, color = "blue") + labs(title = "Simple Linear Regression: MPG vs Weight",

x = "Weight of Car (1000 lbs)", y = "Miles Per Gallon (MPG)")

# --- Multiple Linear Regression ---

multiple\_model <- lm(mpg ~ wt + hp + drat, data = mtcars) # Fit the model summary(multiple\_model) # Summary of the model

# Predictions using the multiple regression model

mtcars$predicted\_mpg <- predict(multiple\_model, newdata = mtcars) # Combine predictions # View the updated dataset with predictions

head(mtcars)

# Plotting actual vs predicted values

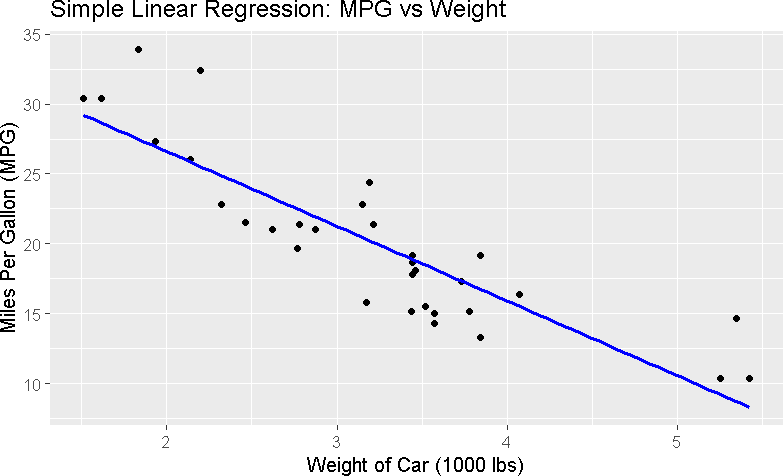
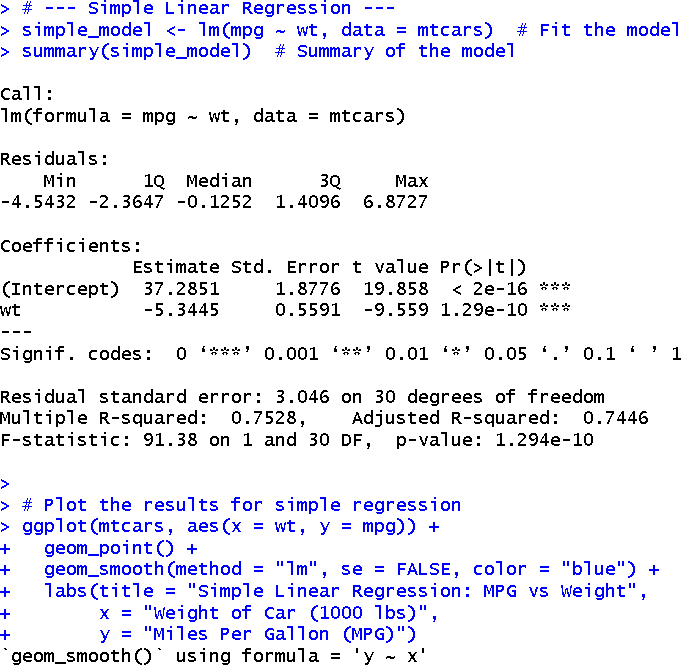
ggplot(mtcars, aes(x = mpg, y = predicted\_mpg)) + geom\_point() +

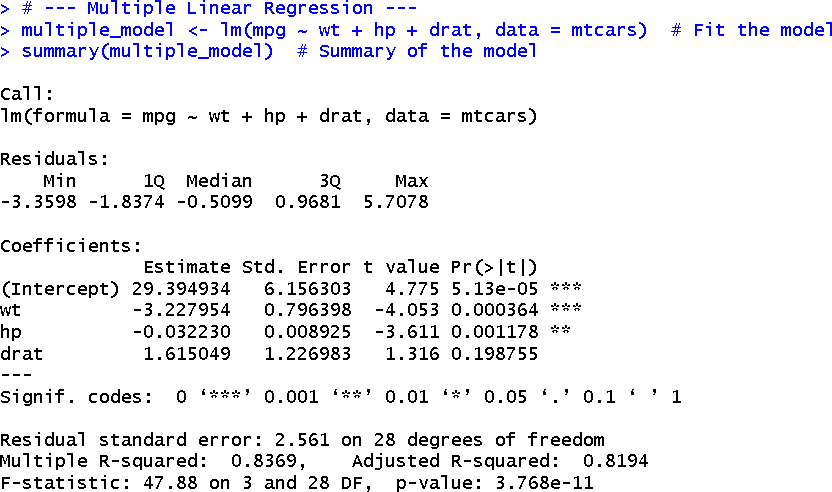
geom\_abline(slope = 1, intercept = 0, color = "red") + labs(title = "Actual vs Predicted MPG",

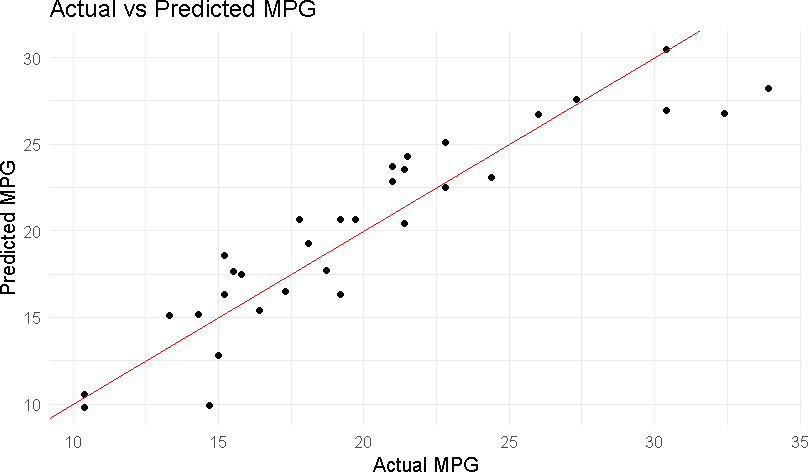
x = "Actual MPG",

y = "Predicted MPG") + theme\_minimal()

### Output:







**Result:**

Thus, the program was executed successfully and the output has been verified.

## EXP NO

**10**

# PERFORM A CHI-SQUARE TEST OF A SAMPLE DATA

## DATE

### Aim:

To perform a Chi-Square Test of a sample data.

### Algorithm:

1. Create Dataset: Construct a matrix to represent the count of preferences for two products among two groups.
2. Assign Row and Column Names: Label the rows as "Group 1" and "Group 2", and columns as "Product A" and "Product B" for clarity.
3. Print Contingency Table: Display the constructed matrix to show the counts of preferences.
4. Perform Chi-Square Test: Use the chisq.test() function to conduct the Chi-Square test on the dataset.
5. Display Test Result: Print the results of the Chi-Square test, including the test statistic and p-value.
6. Interpret Results: Check the p-value against the significance level (0.05) to determine if there is a significant association between groups and product preferences:
   * If p-value < 0.05, reject the null hypothesis.
   * If p-value ≥ 0.05, fail to reject the null hypothesis.

### Program:

# Create a sample dataset

# This dataset represents the count of preferences for two different products among two groups of people data <- matrix(c(30, 10, 20, 40), nrow = 2, byrow = TRUE)

# Assign row and column names for clarity rownames(data) <- c("Group 1", "Group 2") colnames(data) <- c("Product A", "Product B")

# Print the dataset print("Contingency Table:") print(data)

# Perform the Chi-Square test chi\_square\_result <- chisq.test(data)

# Display the result

print("Chi-Square Test Result:") print(chi\_square\_result)

# Interpret the result

if (chi\_square\_result$p.value < 0.05) {

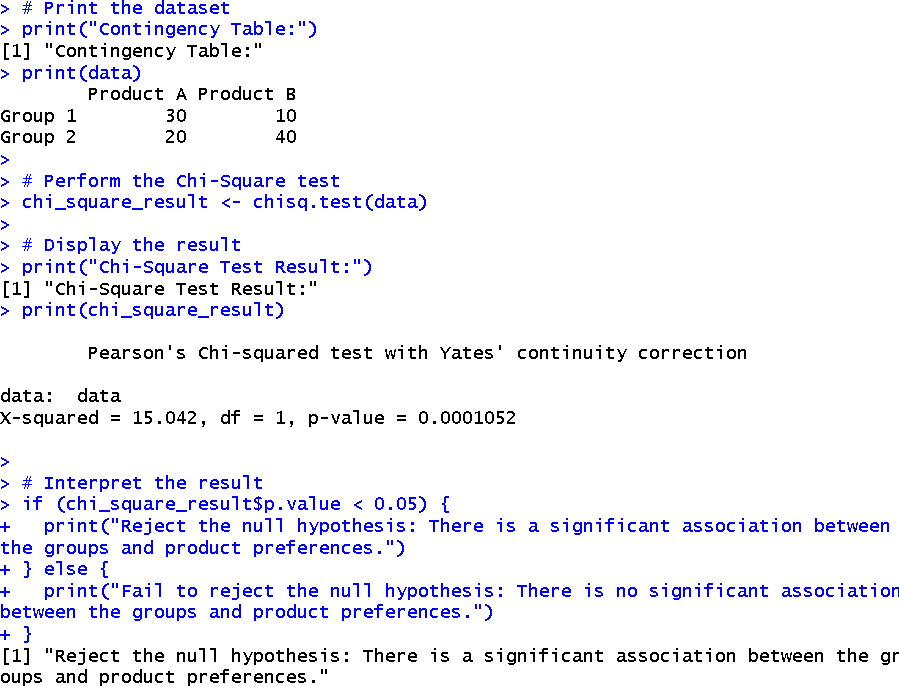
print("Reject the null hypothesis: There is a significant association between the groups and product preferences.")

} else {

print("Fail to reject the null hypothesis: There is no significant association between the groups and product preferences.")

}

### Output:



**Result:**

Thus, the program was executed successfully and the output has been verified.