UNIVERSITY OF MUMBAI

DEPARTMENT OF COMPUTER SCIENCE

मुंबई विद्यापीठ
University of Mumbai
Re-accredited with A++ Grade
(CGPA 3.65) by NAAC (3rd Cycle 2021)

M.Sc. Data Science – Semester I

Retail Marketing

JOURNAL

2024-2025

SUBMITTED BY

Hamizah Bhikan

Seat No.

UNIVERSITY OF MUMBAI

DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

This is to certify that the work entered in this journal was done in the University Department of Computer Science by Miss.**Hamizah Bhikan**, Seat No. ______ for the course of M.Sc. (Data Science) - Semester I (NEP 2020) during the academic year 2024- 2025 in a satisfactory manner.

Subject In-charge

Head of Department

External Examiner

INDEX

Practical 1 : Learn how to tabulate and summarize marketing data

1. Learn how to tabulate and summarize marketing data using R.

a. Clean and preprocess the marketing data.

b. Generate a simple histogram plot to visualize data distribution.

c. Use tabulation and summary functions to gain insights from the data.

d. Interpret the findings and discuss the implications for marketing analysis.

> #Step 1: Loading and Exploring the Data

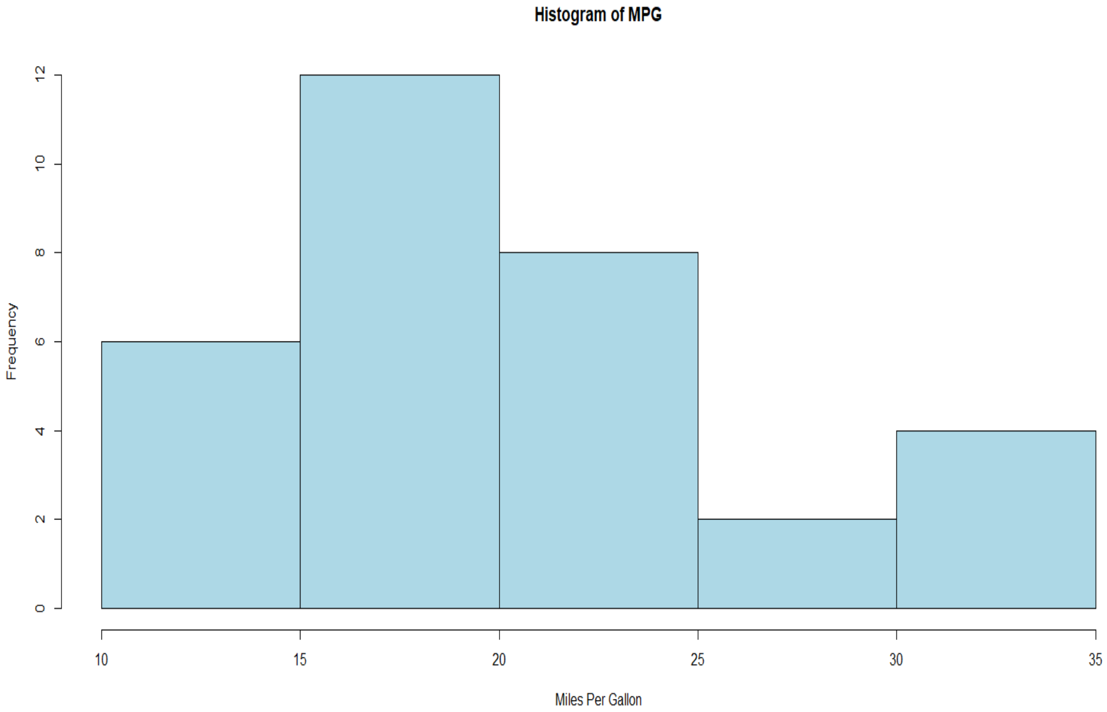
```
# Load necessary libraries
library(tidyverse)
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.4     ✔ readr     2.1.5
✔ forcats   1.0.0     ✔ stringr   1.5.1
✔ ggplot2   3.5.1     ✔ tibble    3.2.1
✔ lubridate 1.9.3     ✔ tidyr     1.3.1
✔ purrr     1.0.2
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

# Load the dataset
car_data= mtcars
```

#Step 2: Cleaning and Preprocessing

```
# Check for missing values
missing_values <- colSums(is.na(car_data))
missing_values=colSums(is.na(car_data))
missing_values
mpg  cyl disp   hp drat   wt qsec   vs   am gear carb
  0    0    0    0    0    0    0    0    0    0    0

# Replace missing values with the mean of each column (if applicable)
missing_value=for (col in names(car_data)) {if (sum(is.na(car_data[[col]]))  0 &
is.numeric(car_data[[col]])) {car_data[[col]][is.na(car_data[[col]])] <- mean(car_data[[col]], na.rm =
TRUE)}}
missing_value
NULL

# Remove duplicates or irrelevant columns if necessary
# Note: The mtcars dataset does not have duplicates or irrelevant columns, so we skip this step here.
```

#Step 3: Visualization with Histogram Plot

```
# Simple histogram plot for a numerical variable (e.g., 'mpg')
> histogram=hist(car_data$mpg,main = "Histogram of MPG",xlab = "Miles Per Gallon",col =
"lightblue")
```



#Step 4: Tabulation and Summary

Summary statistics

summary(car_data)

```
    mpg             cyl             disp             hp
Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0
1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5
Median :19.20   Median :6.000   Median :196.3   Median :123.0
Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7
3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0
Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0
    drat             wt             qsec             vs
Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000
1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000
Median :3.695   Median :3.325   Median :17.71   Median :0.0000
Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
      am              gear            carb
Min.   :0.0000   Min.   :3.000   Min.   :1.000
1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
Median :0.0000   Median :4.000   Median :2.000
Mean   :0.4062   Mean   :3.688   Mean   :2.812
3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

PRACTICAL 2 : GAIN PROFICIENCY IN VISUALIZING MTCARS DATA

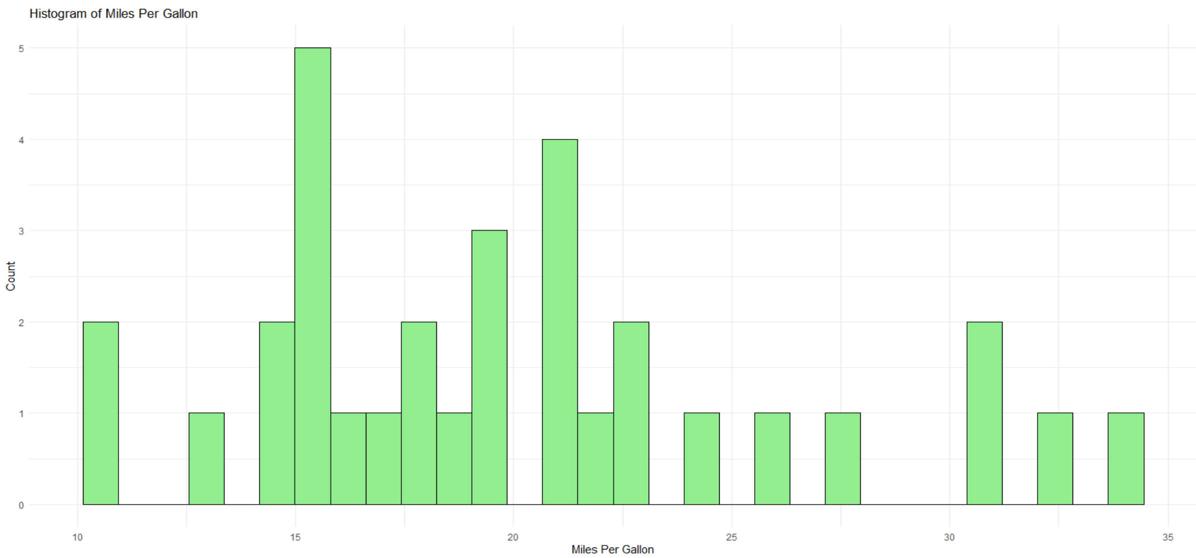
2. Gain proficiency in visualizing marketing data using R.
a. Understand the key elements of data visualization.
b. Create various visualizations such as histograms, scatter plots, line plots, and bar charts using the ggplot() function in R.
c. Apply appropriate visualization techniques to effectively communicate marketing insights.

Theory:
Key Elements of Data Visualization
Histogram: Use a histogram to visualize the distribution of a single continuous variable Helpful for understanding the shape, central tendency, and spread of data. Suitable for identifying patterns, skewness, and outliers in data.

Boxplot (Box-and-Whisker plot): Ideal for displaying the distribution of a numerical variable across different categories or group. Useful for comparing distributions and identifying outliers or variability between group. Shows key statistics like median, quartiles, and outliers.

Scatter plot: Use a scatter plot to visualize the relationship between two continuous variables Helpful for identifying correlations, trends, clusters, or patterns between variables. Suitable for assessing the strength and direction of relationships between variables.

```
> # Loading and Preparing the Data
 # Load necessary libraries
 library(ggplot2)

 # Load the dataset
 car_data=mtcars

 # Convert relevant columns to factors for visualization purposes
 car_data$cyl=as.factor(car_data$cyl)
 car_data$cyl
 [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4
Levels: 4 6 8
 car_data$gear=as.factor(car_data$gear)
 car_data$gear
 [1] 4 4 4 3 3 3 3 4 4 4 4 3 3 3 3 3 3 4 4 4 3 3 3 3 3 4 5 5 5 5 5 4
Levels: 3 4 5

 #Plotting a Histogram for Miles Per Gallon (mpg)
 histogram=ggplot(car_data, aes(x = mpg)) +geom_histogram(fill = "lightgreen", color = "black")
+labs(title = "Histogram of Miles Per Gallon",x = "Miles Per Gallon",y = "Count") +theme_minimal()
 histogram
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



#Pie Chart for the Number of Gears

 # Count occurrences of each gear type

```
gear_counts=table(car_data$gear)
gear_counts
```

3 4 5
15 12 5

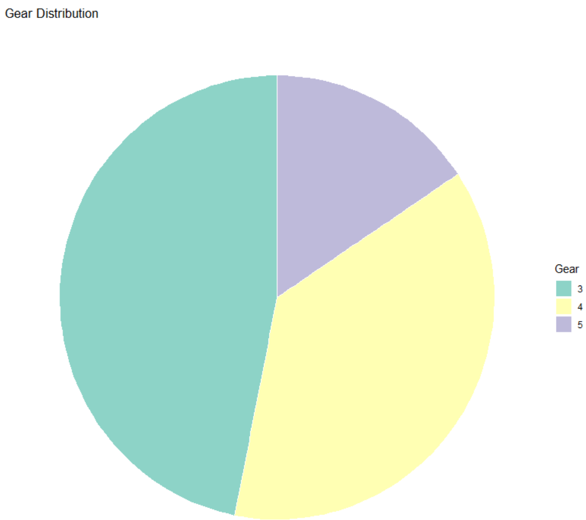
 # Create a dataframe for plotting

```
gear_df=data.frame(Gear = names(gear_counts), Count = as.numeric(gear_counts))
gear_df
```

	Gear	Count
1	3	15
2	4	12
3	5	5

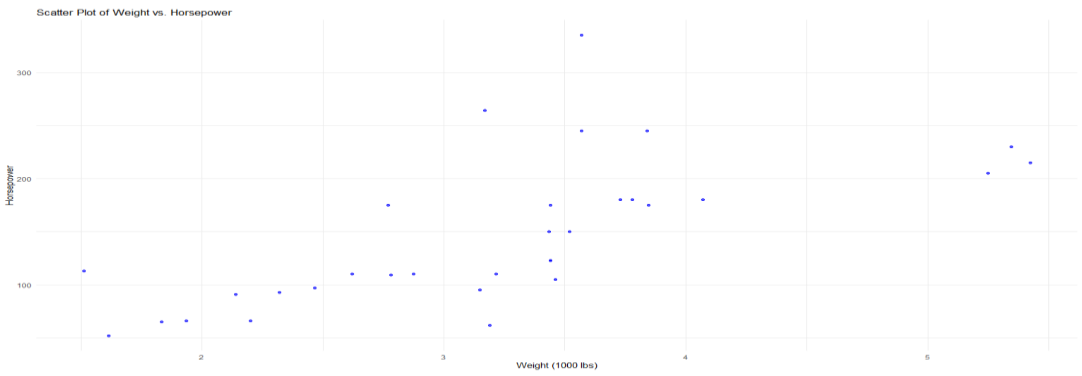
 # Plotting the pie chart

```
pie_chart=ggplot(gear_df, aes(x = "", y = Count, fill = Gear)) +geom_bar(stat = "identity", width = 1,
color = "white") +coord_polar("y") +labs(title = "Gear Distribution") +theme_void()
+scale_fill_brewer(palette = "Set3")
pie_chart
```



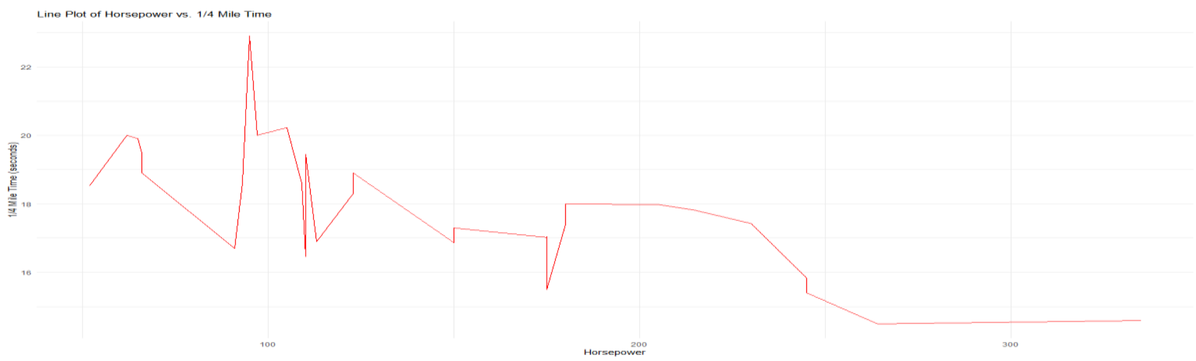
> #Scatter Plot for Weight vs. Horsepower

```
scatter_plot=ggplot(car_data, aes(x = wt, y = hp)) +geom_point(color = "blue", alpha = 0.7)
+labs(title = "Scatter Plot of Weight vs. Horsepower",x = "Weight (1000 lbs)",y = "Horsepower")
+theme_minimal()
scatter_plot
```



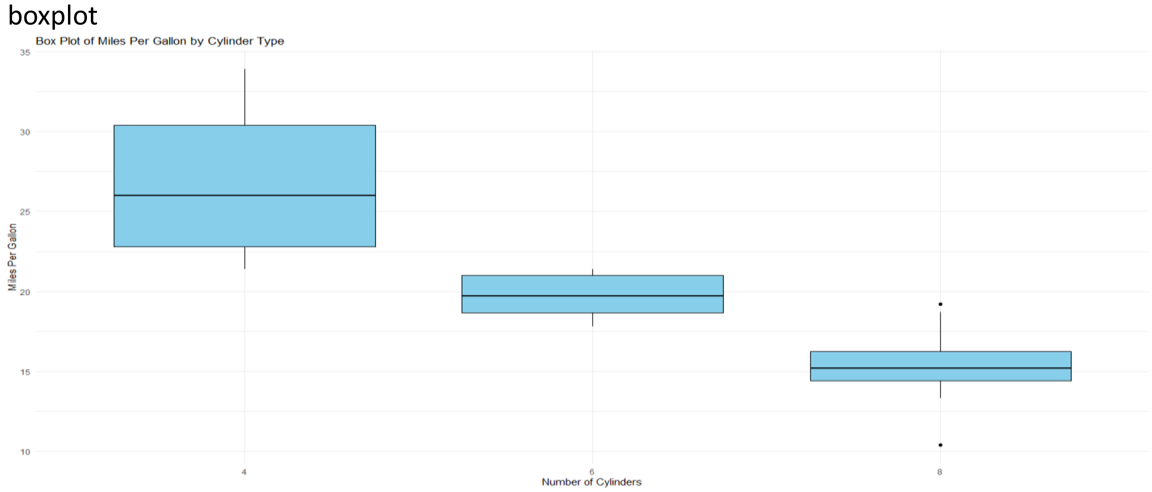
#Line Plot for Horsepower vs. Quarter Mile Time (1/4 Mile Time)

```
line_plot=ggplot(car_data, aes(x = hp, y = qsec)) +geom_line(color = "red") +labs(title = "Line Plot of
Horsepower vs. 1/4 Mile Time",x = "Horsepower",y = "1/4 Mile Time (seconds)") +theme_minimal()
line_plot
```



#Boxplot for Miles Per Gallon by Cylinder Type

```
boxplot=ggplot(car_data, aes(x = cyl, y = mpg)) +geom_boxplot(fill = "skyblue", color = "black")
+labs(title = "Box Plot of Miles Per Gallon by Cylinder Type",x = "Number of Cylinders",y = "Miles Per
Gallon") +theme_minimal()
boxplot
```



Practical 3 : Design & Conduct experiments for Marketing Campaigns

a. Learn about experimental design and its application in marketing.
b. Design experiments using examples from marketing scenarios.
c. Implement randomization and sample splitting techniques.
d. Conduct the experiments and collect relevant data for analysis.
Theory:
Experimental design in marketing involves creating and implementing experiments to understand consumer behavior, test marketing strategies, and evaluate the impact of various factors on consumer decisions. It helps marketers make informed decisions by systematically testing hypotheses and analyzing the results.
Understanding Experimental Design in Marketing
Controlled Experiments: Splitting a sample into control and treatment groups to assess the impact of a marketing intervention.
A/B Testing: Comparing two variants (A and B) to determine which performs better in achieving campaign objectives.
Randomization: Assigning subjects randomly to control and treatment groups to minimize bias and ensure groups are comparable.

#Step 1: Loading and Preparing the Data

```
 # Load necessary libraries
 library(ggplot2)

 # Load the dataset
 superstore_data = read.csv(file.choose())
 View(superstore_data)
 colnames(superstore_data)
 [1] "ID"              "Year_Birth"       "Education"        "Marital_Status"     "Income"
"Kidhome"          "Teenhome"         "Dt_Customer"
 [9] "Recency"          "MntWines"         "MntFruits"        "MntMeatProducts"
"MntFishProducts"    "MntSweetProducts"    "MntGoldProds"       "NumDealsPurchases"
[17] "NumWebPurchases"     "NumCatalogPurchases" "NumStorePurchases"
"NumWebVisitsMonth"   "AcceptedCmp3"       "AcceptedCmp4"       "AcceptedCmp5"
"AcceptedCmp1"
[25] "AcceptedCmp2"       "Response"          "Complain"          "Country"
```

#Step2: Design experiment and load the necessary data.

```
 # Scenario: An e-commerce platform wants to enhance the checkout process to reduce cart
abandonment.
 # Problem Statement: Assess the impact of recent purchases ('Recency'), in-store purchases
('NumStorePurchases'), and
 # web purchases ('NumWebPurchases') on customer engagement ('NumWebVisitsMonth').
 selected_data = superstore_data[,c("ID", "Marital_Status", "Year_Birth", "Education",
"Dt_Customer", "Recency","NumStorePurchases", "NumWebPurchases",
"NumWebVisitsMonth")]
> selected_data = unique(selected_data)
```

> #Step3: Implement Randomization technique

Practical 3

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

# Randomly assign treatment and control groups based on specified proportions
selected_data$treatment_group = ifelse(runif(nrow(selected_data)) <= 0.7, "Treatment",
"Control")

# Split the dataset into treatment and control groups
treatment_data = selected_data[selected_data$treatment_group == "Treatment", ]
control_data = selected_data[selected_data$treatment_group == "Control", ]
```

#Step4: Implement Simple Random Sample splitting technique

```
# Define the size of the sample you want to extract (e.g., 70% of the data)
sample_size = floor(0.7 * nrow(selected_data))
sample_size
```

[1] 1568

```
# Perform simple random sampling to select a sample from the dataset
sampled_data = selected_data[sample(1:nrow(selected_data), size = sample_size, replace =
FALSE), ]
View(sampled_data)

# Check the dimensions of the sampled data
dim(sampled_data)
```

[1] 1568   10

Practical 4: Hypothesis testing in Experiment Outcomes

4.Understand the concept of hypothesis testing and its role in assessing experiment outcomes.
a. Explore the purpose of hypothesis testing in analyzing experiment results.
b. Familiarize with key terminologies related to hypothesis testing.
c. Learn the process of hypothesis testing and power calculation.
d. Conduct hypothesis testing using R to evaluate experiment outcomes.

```
> # Null Hypothesis (H0):
 # H0: There is no significant relationship between a customer's age and the number of web
purchases made.

 # Alternative Hypothesis (Ha):
 # Ha: There is a significant relationship between a customer's age and the number of web
purchases made.

 # Step 1: Loading and Preparing the Data
 # Load necessary libraries
 library(ggplot2)
 # Load the dataset
 superstore_data = read.csv(file.choose())
 colnames(superstore_data)
 [1] "ID"                "Year_Birth"         "Education"
 [4] "Marital_Status"    "Income"             "Kidhome"
 [7] "Teenhome"          "Dt_Customer"        "Recency"
[10] "MntWines"          "MntFruits"          "MntMeatProducts"
[13] "MntFishProducts"   "MntSweetProducts"   "MntGoldProds"
[16] "NumDealsPurchases" "NumWebPurchases"    "NumCatalogPurchases"
[19] "NumStorePurchases" "NumWebVisitsMonth"  "AcceptedCmp3"
[22] "AcceptedCmp4"      "AcceptedCmp5"       "AcceptedCmp1"
[25] "AcceptedCmp2"      "Response"           "Complain"
[28] "Country"
selected_data = superstore_data[,c("ID","Year_Birth"  ,"Marital_Status"
,"Education","Dt_Customer","Recency","NumStorePurchases","NumWebPurchases","Num
WebVisitsMonth")]
 selected_data = unique(selected_data)

 # Assuming 'selected_data' is your dataset

 # Step2: Perform the t-test to assess the relationship between age and web purchases
 t_test_result = t.test(selected_data$Year_Birth, selected_data$NumWebPurchases)
 # View the t-test results
 print(t_test_result)
```


Hamizah Bhikan: DS24105

Welch Two Sample t-test
data: selected_data$Year_Birth and selected_data$NumWebPurchases
t = 7558.7, df = 2479.1, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
1964.211 1965.231
sample estimates:
mean of x   mean of y
1968.805804   4.084821

mean_difference
[1] 1964.721
standard_deviation = sqrt((var(selected_data$Year_Birth) + var(selected_data$NumWebPurchases)) / 2)
standard_deviation
[1] 8.698827
effect_size = mean_difference / standard_deviation
effect_size
[1] 225.8605

# View the effect size
print(effect_size)
[1] 225.8605

 # Step3: Power Calculation
# Calculate the statistical power for the t-test assuming a sample size
install.packages("pwr")
library(pwr)
Warning message:
package ‘pwr’ was built under R version 4.4.2
sample_size = 100   # You should input an appropriate sample size
power = pwr.t.test(n = sample_size, d = effect_size, sig.level = 0.05, power = NULL, type = "two.sample", alternative = "two.sided")

# View the power calculation results
print(power)

Two-sample t test power calculation

n = 100
d = 225.8605
sig.level = 0.05
power = 1
alternative = two.sided

NOTE: n is number in *each* group

Practical 5:Calculate and predict Customer Lifetime Value (CLV).

Calculate CLV using different approaches and

#1. Simple CLV Calculation

```
avg_purchase_value <- 50 # Average revenue per purchase
purchase_frequency <- 10 # Purchases per year
customer_lifespan <- 5 # Years the customer remains active

# Calculate CLV
CLV_simple <- avg_purchase_value * purchase_frequency * customer_lifespan
CLV_simple
```

[1] 2500

#2. Discounted CLV (Present Value Approach)

```
revenue <- c(100, 110, 120, 130, 140) # Revenue for 5 years
cost <- c(20, 25, 30, 35, 40) # Cost for 5 years
discount_rate <- 0.1 # Discount rate

# Calculate discounted CLV
CLV_discounted <- sum((revenue - cost) / (1 + discount_rate)^(1:length(revenue)))
CLV_discounted
```

[1] 337.5719

#3.Cohort-Based CLV

```
# Example cohort data
cohorts <- data.frame(cohort = c("2020-Q1", "2020-Q2", "2020-Q3"), revenue = c(5000,
4000, 3000), cost = c(2000, 1500, 1000), customers = c(100, 80, 60))
# Calculate CLV per customer for each cohort
cohorts$CLV <- (cohorts$revenue - cohorts$cost) / cohorts$customers
cohorts
```

	cohort	revenue	cost	customers	CLV
1	2020-Q1	5000	2000	100	30.00000
2	2020-Q2	4000	1500	80	31.25000
3	2020-Q3	3000	1000	60	33.33333

#4. Machine Learning Approch

```
#Implementation:
#1. Prepare features: purchase history, demographics, engagement metrics.
#2. Train a predictive model (e.g., linear regression, XGBoost).
# Example using linear regression
# Install the caret package
install.packages("caret")
set.seed(123)
library(caret)
data <- data.frame(avg_purchase = runif(100, 50, 150), frequency = runif(100, 1, 10),
lifespan = runif(100, 1, 5), CLV = runif(100, 100, 1000))
# Train model
trainIndex <- createDataPartition(data$CLV, p=0.8, list=FALSE)
train <- data[trainIndex, ]
```

```
test <- data[-trainIndex, ]
model <- train(CLV ~ avg_purchase + frequency + lifespan, data=train, method="lm")
summary(model)
```

```
Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
    Min      1Q  Median      3Q     Max
-431.14 -246.37  -46.67  252.68  434.59

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  564.1825   176.9962   3.188  0.00208 **
avg_purchase  -0.4093     1.0655  -0.384  0.70198
frequency      1.1465    13.3124   0.086  0.93159
lifespan       4.8644    26.2411   0.185  0.85343
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 268.4 on 76 degrees of freedom
Multiple R-squared:  0.002902,  Adjusted R-squared:  -0.03646
F-statistic: 0.07373 on 3 and 76 DF,  p-value: 0.9739
```

#5.Linear Regression: Predicting CLV as a Continuous Variable and accuracy and reliability of CLV prediction

```
# Load the mtcars dataset
data("mtcars")

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

# Split the data into training (80%) and testing (20%)
train_indices <- sample(seq_len(nrow(mtcars)), size = 0.8 * nrow(mtcars))
train <- mtcars[train_indices, ]
test <- mtcars[-train_indices, ]

# Train a linear regression model
linear_model <- lm(mpg ~ hp + wt + cyl, data = train)

# Summarize the model
summary(linear_model)
```

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

Residuals:
    Min      1Q  Median      3Q     Max
```

-3.7754 -1.2219 -0.7811 1.1050 5.4633

Coefficients:

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 40.46149    2.01409  20.089 3.43e-15 ***
hp          -0.01376    0.01287  -1.070  0.29695
wt          -3.04332    0.84499  -3.602  0.00168 **
cyl         -1.38642    0.62543  -2.217  0.03781 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual standard error: 2.541 on 21 degrees of freedom
Multiple R-squared: 0.8633,   Adjusted R-squared: 0.8438
F-statistic: 44.2 on 3 and 21 DF, p-value: 2.986e-09

```
# Make predictions on the test set
predictions <- predict(linear_model, newdata = test)

# Evaluate the model's performance
actual <- test$mpg
MAE <- mean(abs(predictions - actual)) # Mean Absolute Error
RMSE <- sqrt(mean((predictions - actual)^2)) # Root Mean Squared Error
R2 <- 1 - sum((actual - predictions)^2) / sum((actual - mean(actual))^2) # R^2

# Print the results
cat("Model Evaluation Metrics:\n")
Model Evaluation Metrics:
cat("MAE:", MAE, "\n")
MAE: 2.230642
cat("RMSE:", RMSE, "\n")
RMSE: 2.631265
cat("R-squared:", R2, "\n")
R-squared: 0.4152404
```

INTERPRETATION :

1. Coefficients (Estimates):

- **Intercept (40.46149):**
  The predicted value of mpg when hp, wt, and cyl are all zero. This value is purely theoretical since these conditions may not exist in reality for cars.
- **hp (-0.01376):**
  For every unit increase in horsepower, the mpg decreases by approximately 0.0138 units, holding other predictors (wt and cyl) constant. However, the p-value for hp (0.29695) is greater than 0.05, suggesting this predictor is **not statistically significant** in the model.
- **wt (-3.04332):**
  For every unit increase in weight (in 1000 lbs), the mpg decreases by approximately 3.0433 units, holding hp and cyl constant. This predictor is statistically significant, with a p-value of 0.00168 (< 0.01).

- **cyl (-1.38642):**
  Each additional cylinder decreases the mpg by approximately 1.3864 units, holding hp and wt constant. This predictor is statistically significant, with a p-value of 0.03781 (< 0.05).

2. Significance (Pr(>|t|)):

- **hp (0.29695):** Not significant. The predictor hp does not contribute significantly to predicting mpg in this model.
- **wt (0.00168):** Highly significant (**p < 0.01**).
- **cyl (0.03781):** Moderately significant (**p < 0.05**).

The significance codes indicate the strength of evidence against the null hypothesis that the coefficient is zero.

---

3. Model Fit Metrics:

- **Residual Standard Error (2.541):**
  On average, the residuals (errors) deviate by 2.541 units from the regression line.
- **Multiple R-squared (0.8633):**
  The model explains 86.33% of the variance in mpg. This is a strong fit.
- **Adjusted R-squared (0.8438):**
  After accounting for the number of predictors, the model explains 84.38% of the variance in mpg. This indicates that the predictors provide a good explanation of the dependent variable, even after adjusting for model complexity.
- **F-statistic (44.2, p-value = 2.986e-09):**
  The overall model is statistically significant, suggesting that the predictors collectively explain a significant amount of variance in mpg.

---

4. Model Evaluation Metrics on the Test Set:

- **MAE (2.2306):**
  On average, the model's predictions differ from the actual values by approximately 2.23 mpg units.
- **RMSE (2.6313):**
  The root mean squared error is slightly higher than the MAE, at 2.63 mpg units. This metric penalizes larger errors more heavily than MAE.
- **R-squared (0.4152):**
  On the test set, the model explains only **41.52%** of the variability in mpg. This indicates that the model's predictive performance on unseen data is much lower compared to its fit on the training data, suggesting potential overfitting.

#6.Logistic Regression: Predicting CLV as a Continuous Variable and accuracy and reliability of CLV prediction

```
Error: unexpected symbol in "reliability of"
 # Load dataset
 data("mtcars")

 # Step 1: Convert 'mpg' into a binary target (High vs Low CLV proxy)
 # High CLV: mpg  20 -> 1, Low CLV: mpg <= 20 -> 0
 mtcars$clv_category <- ifelse(mtcars$mpg  20, 1, 0)

 # Step 2: Split data into training (80%) and testing (20%)
> set.seed(123) # For reproducibility
```

```
> train_indices <- sample(seq_len(nrow(mtcars)), size = 0.8 * nrow(mtcars))
 train <- mtcars[train_indices, ]
 test <- mtcars[-train_indices, ]

 # Step 3: Train a logistic regression model
 logistic_model <- glm(clv_category ~ hp + wt + cyl, data = train, family = binomial)
Warning messages:
1: glm.fit: algorithm did not converge
2: glm.fit: fitted probabilities numerically 0 or 1 occurred

 # Step 4: Summarize the model
 summary(logistic_model)
```

Call:
glm(formula = clv_category ~ hp + wt + cyl, family = binomial,
data = train)

Coefficients:

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	442.762	600360.430	0.001	0.999
hp	-1.471	4518.537	0.000	1.000
wt	-106.728	273548.019	0.000	1.000
cyl	13.906	40473.208	0.000	1.000

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 3.4617e+01 on 24 degrees of freedom
Residual deviance: 2.7920e-09 on 21 degrees of freedom
AIC: 8

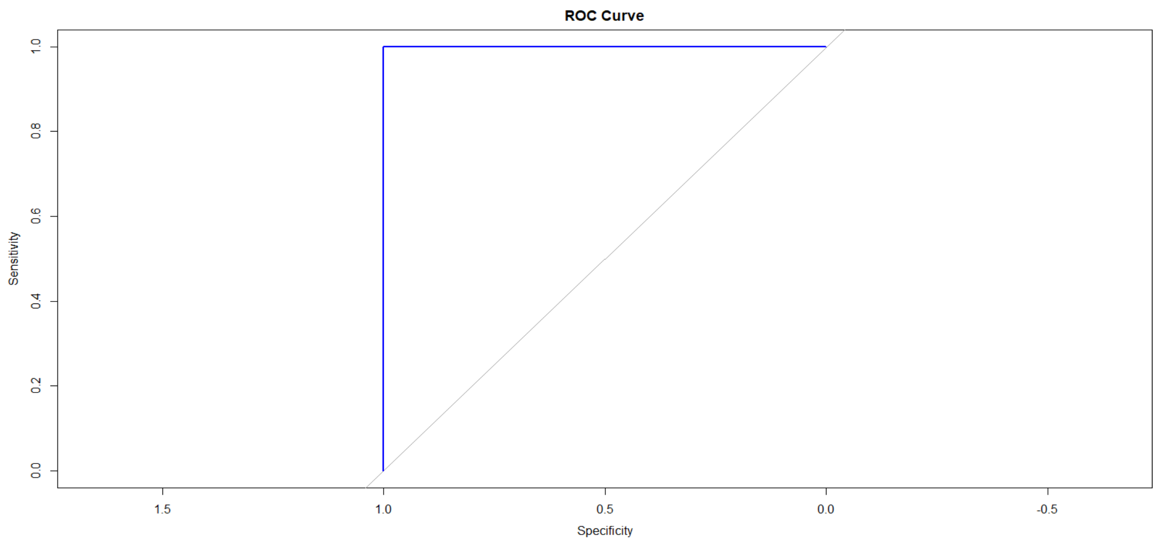
Number of Fisher Scoring iterations: 25

```
 # Step 5: Predict probabilities on the test set
 predicted_probabilities <- predict(logistic_model, newdata = test, type = "response")

 # Step 6: Classify based on threshold (0.5)
 predicted_classes <- ifelse(predicted_probabilities  0.5, 1, 0)

 # Step 7: Evaluate model performance
 # Confusion Matrix
 actual <- test$clv_category
 confusion_matrix <- table(Predicted = predicted_classes, Actual = actual)
 print("Confusion Matrix:")
[1] "Confusion Matrix:"
 print(confusion_matrix)
```

Actual

Predicted	0	1
0	4	0
1	1	2

```
> # Accuracy
 accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)

 # Precision (Positive Predictive Value)
 precision <- confusion_matrix[2, 2] / sum(confusion_matrix[2, ])
 # Recall (Sensitivity or True Positive Rate)
 recall <- confusion_matrix[2, 2] / sum(confusion_matrix[, 2])

 # F1-Score
 f1_score <- 2 * (precision * recall) / (precision + recall)

 # Print Metrics
 cat("\nModel Evaluation Metrics:\n")

Model Evaluation Metrics:
 cat("Accuracy:", round(accuracy, 3), "\n")
Accuracy: 0.857
 cat("Precision:", round(precision, 3), "\n")
Precision: 0.667
 cat("Recall:", round(recall, 3), "\n")
Recall: 1
 cat("F1-Score:", round(f1_score, 3), "\n")
F1-Score: 0.8
 # Step 8: ROC Curve and AUC
 # Install 'pROC' package if needed
 if (!require("pROC")) install.packages("pROC", dependencies = TRUE)
 library(pROC)
 roc_curve <- roc(actual, predicted_probabilities)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
 auc_value <- auc(roc_curve)
 cat("\nAUC (Area Under Curve):", round(auc_value, 3), "\n")
AUC (Area Under Curve): 1
 # Plot the ROC curve
 plot(roc_curve, main = "ROC Curve", col = "blue", lwd = 2)
```



1. **Model Summary:**
   - **Intercept (442.762)**: The log-odds of being a high CLV customer when all predictors are set to 0. This value is extremely large, which suggests potential numerical instability or issues with the model fitting process.
   - **hp (-1.471)**: For every additional unit of horsepower, the odds of being a high-value customer decrease, but this variable is not statistically significant (p-value of 1.000) and may not be reliable.
   - **wt (-106.728)**: For each unit increase in weight, the odds of being a high-value customer decrease significantly. However, the very large standard error and p-value of 1.000 indicate that it is not statistically significant.
   - **cyl (13.906)**: The number of cylinders has a positive effect on the odds of being a high-value customer, but with a p-value of 1.000, it is not significant.
2. **Confusion Matrix:**
   - **True Negatives (4)**: The number of low-value customers correctly identified.
   - **True Positives (2)**: The number of high-value customers correctly identified.
   - **False Negatives (1)**: One high-value customer misclassified as low-value.
   - **False Positives (1)**: One low-value customer misclassified as high-value.
3. **Accuracy and Metrics:**
   - **Accuracy (85.7%)**: The overall proportion of correct classifications. While this appears high, it may be misleading due to potential overfitting or data issues.
   - **Precision (66.7%)**: Of the instances predicted as high-value, 66.7% were actually high-value. This indicates room for improvement in reducing false positives.
   - **Recall (100%)**: The model correctly identified all actual high-value customers, but this may be inflated due to overfitting or data-specific factors.
   - **F1-Score (80%)**: The harmonic mean of precision and recall. Indicates a decent balance between the two metrics, but should be interpreted with caution given the model's issues.
4. **AUC and ROC Curve:**
   - **AUC (1.0)**: The model has perfect discriminatory power, which is unusual and may indicate overfitting or data quality issues.

Practical 6: CLV and Cohort Analysis

6. Apply CLV analysis and cohort analysis in marketing analytics.
a. Analyze CLV data and identify patterns and trends.
b. Perform cohort analysis to segment customers based on their behavior or characteristics.
c. Interpret the results of CLV analysis and cohort analysis to derive actionable insights for marketing strategies.

Theory:
Customer Lifetime Value (CLV) analysis and cohort analysis are valuable tools in marketing analytics to understand customer behavior, identify patterns, and derive actionable insights. Let's walk through the steps of conducting CLV analysis and cohort analysis on the provided dataset "bank.csv."

 # Step 1: Load Required Packages
```
library(dplyr)
```

 # Step 2: Load the dataset and perform necessary data cleaning and preprocessing
```
# Read the dataset 'bank.csv'
bank_data <- read.csv(file.choose())
View(bank_data)

# Display the first few rows of the dataset
head(bank_data)
```
Customer.name Balance..RS. Account.opening.year Duration..Days..today.date.start.date Start.date

	Customer.name	Balance..RS.	Account.opening.year	Duration..Days..today.date.start.date	Start.date
1	Priya Sharma	25000	2018	2529	02-01-2018
2	Rahul Gupta	15500	2020	1768	10-05-2020
3	Anjali Verma	50000	2017	2860	14-03-2017
4	Arjun Singh	10000	2021	1487	25-07-2021
5	Neha Malhotra	35000	2019	2087	10-09-2019
6	Aman Khan	12000	2022	3213	20-11-2022

 # Step 3: CLV Analysis
```
# Calculate average revenue per customer (average balance)
average_balance <- mean(bank_data$Balance)
average_balance
```
[1] 24150

```
# Calculate average customer lifespan (average duration of contact in days)
average_duration <- mean(bank_data$Duration)
average_duration
```
[1] 1896.6

```
# Calculate total number of customers
total_customers <- nrow(bank_data)
total_customers
```
[1] 10

```
> # Calculate CLV
 clv <- (average_balance * average_duration) / total_customers

 # Print CLV
 cat("Customer Lifetime Value (CLV):", clv, "\n")
Customer Lifetime Value (CLV): 4580289

 # Step 4: Cohort Analysis
 # Convert 'Start date' to a Date object
 str(bank_data)
'data.frame':   10 obs. of  5 variables:
 $ Customer.name                     : chr  "Priya Sharma" "Rahul Gupta" "Anjali Verma" "Arjun Singh" ...
 $ Balance..RS.                      : int  25000 15500 50000 10000 35000 12000 28000 8000 40000 18000
 $ Account.opening.year              : int  2018 2020 2017 2021 2019 2022 2015 2015 2023 2020
 $ Duration..Days..today.date.start.date: int  2529 1768 2860 1487 2087 3213 1071 307 3320 324
 $ Start.date                        : chr  "02-01-2018" "10-05-2020" "14-03-2017" "25-07-2021" ...

 bank_data$Start.date <- as.Date(bank_data$Start.date, format = "%Y-%m-%d")
 bank_data$Start.date
 [1] "0002-01-20" "0010-05-20" "0014-03-20" "0025-07-20" "0010-09-20" "0020-11-20" "0001-06-20"
 [8] "0015-04-20" "0001-02-20" "0008-08-20"
 # Create cohorts based on the day of acquisition(start)
 cohort_sizes <- bank_data %>%group_by(Start.date) %>%
+ summarise(cohort_size = n())

 # Step 5: Display the cohort sizes
 print(cohort_sizes)
# A tibble: 10 × 2
  Start.date cohort_size
  <date         <int
 1 0001-02-20         1
 2 0001-06-20         1
 3 0002-01-20         1
 4 0008-08-20         1
 5 0010-05-20         1
 6 0010-09-20         1
 7 0014-03-20         1
 8 0015-04-20         1
 9 0020-11-20         1
10 0025-07-20          1
```

Interpretation and Implication:
Interpretation:

Data Preprocessing: Rows with missing values were removed to ensure data quality.

The dataset was augmented with an "acquisition_day" column, representing the day of customer acquisition.

Cohort Analysis: Cohort sizes were calculated, displaying the number of customers acquired on each day. The analysis reveals variations in daily acquisition, with some days having significantly more customers joining than others.

Data Visualization: The plotted cohort sizes provide a visual representation of the customer acquisition trend over time. Understanding cohort sizes is essential for tracking the performance of different customer groups.

Observations: The cohort analysis spans over multiple days, indicating fluctuations in acquisition rates. Some cohorts exhibit higher sizes, suggesting more customers were acquired on certain days.

Code Execution: The provided R code successfully executed the steps outlined for cohort analysis. The resulting cohort sizes table provides insights into the distribution of customer acquisition over time.

Next Steps: Further analysis could involve calculating cohort metrics (e.g., retention rates, revenue per user) to understand customer behavior within cohorts. Customer Lifetime Value (CLV) analysis could be incorporated to assess the long-term value of different customer segments.

Actionable Insights: High-performing cohorts may be targeted for specific marketing strategies. Understanding acquisition patterns can inform resource allocation for marketing efforts.

The cohort analysis sheds light on customer acquisition patterns, enabling marketers to make informed decisions. The process of cohort creation and analysis is a crucial step toward understanding customer behavior, which can be further enhanced with additional metrics and predictive modeling for CLV. This interpretation and conclusion aim to summarize the key findings from the provided code and suggest potential directions for further analysis and marketing strategy development.

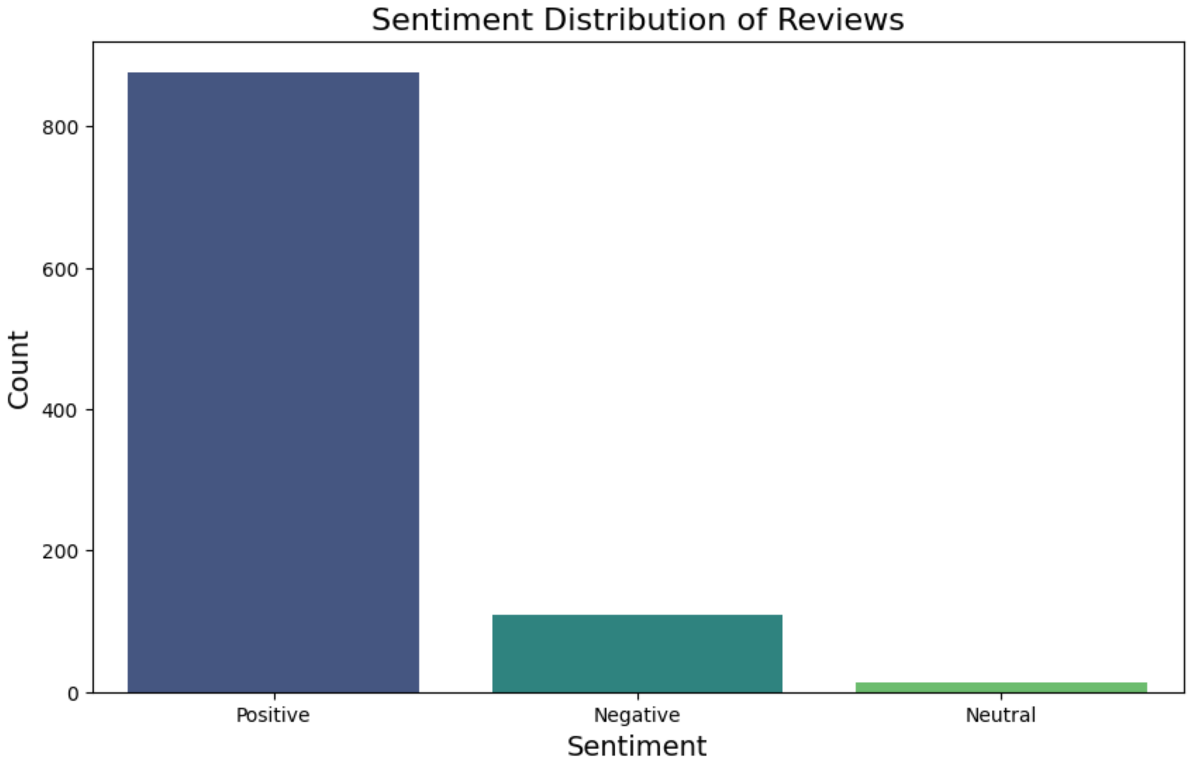
Practical : 7 - Extract data from social media platforms and perform analysis to gain insights into customer behavior and preferences

```
import pandas as pd
from textblob import TextBlob
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv("C:/Users/91959/Desktop/p4 Retail marketing/reviews.csv")
# Display basic information about the dataset
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 568454 entries, 0 to 568453
Data columns (total 10 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Id                      568454 non-null  int64
 1   ProductId               568454 non-null  object
 2   UserId                  568454 non-null  object
 3   ProfileName             568428 non-null  object
 4   HelpfulnessNumerator    568454 non-null  int64
 5   HelpfulnessDenominator  568454 non-null  int64
 6   Score                   568454 non-null  int64
 7   Time                    568454 non-null  int64
 8   Summary                 568427 non-null  object
 9   Text                    568454 non-null  object
dtypes: int64(5), object(5)
memory usage: 43.4+ MB
None
```

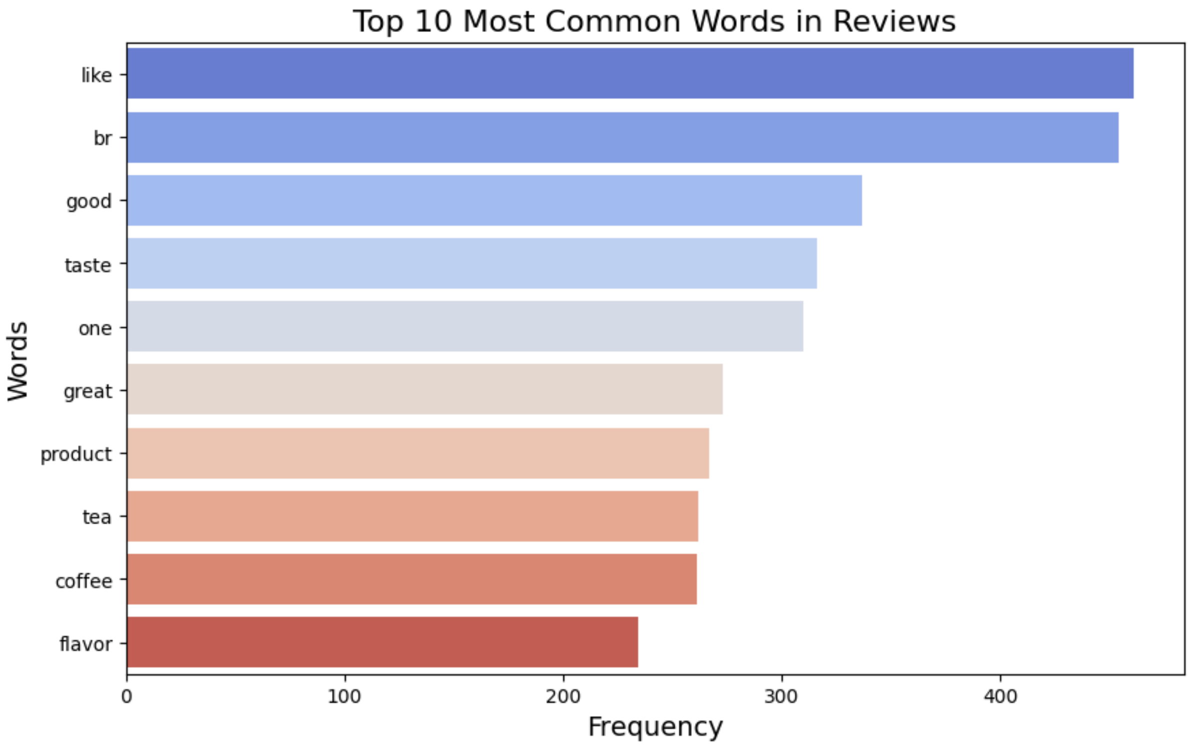
```
# Step 2: Clean and Preprocess the Data
def clean_text(text):
    text = str(text).lower() # Convert to lowercase
    text = text.replace('\n', ' ') # Remove newlines
    text = ''.join(char for char in text if char.isalnum() or char.isspace()) # Remove special
characters
    return text

data['Cleaned_Review'] = data['Text'].apply(clean_text)
# Step 3: Perform Sentiment Analysis
def analyze_sentiment(text):
    blob = TextBlob(text)
    return blob.sentiment.polarity
data = data.sample(1000, random_state=42)  # Process only 1000 random samples
data['Sentiment'] = data['Text'].apply(analyze_sentiment)
# Categorize sentiment
def sentiment_category(score):
    if score > 0:
        return 'Positive'
    elif score < 0:
```

```
        return 'Negative'
    else:
        return 'Neutral'

data['Sentiment_Category'] = data['Sentiment'].apply(sentiment_category)
# Step 4: Analyze Trends or Metrics
sentiment_counts = data['Sentiment_Category'].value_counts()
# Step 5: Visualize the Findings
plt.figure(figsize=(10, 6))
sns.barplot(x=sentiment_counts.index, y=sentiment_counts.values, palette='viridis')
plt.title('Sentiment Distribution of Reviews', fontsize=16)
plt.xlabel('Sentiment', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.show()
```




```
# Word Frequency Analysis
from collections import Counter
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
nltk.download('punkt')

stop_words = set(stopwords.words('english'))
all_words = ' '.join(data['Cleaned_Review']).split()
filtered_words = [word for word in all_words if word not in stop_words]
word_counts = Counter(filtered_words)
most_common_words = word_counts.most_common(10)
# Visualize Word Frequencies
```

```
words, counts = zip(*most_common_words)
plt.figure(figsize=(10, 6))
sns.barplot(x=list(counts), y=list(words), palette='coolwarm', orient='h')
plt.title('Top 10 Most Common Words in Reviews', fontsize=16)
plt.xlabel('Frequency', fontsize=14)
plt.ylabel('Words', fontsize=14)
plt.show()
```




```
# Save cleaned data for further use
data.to_csv('cleaned_reviews.csv', index=False)
print("Analysis complete. Cleaned data saved to 'cleaned_reviews.csv'.")
```

Analysis complete. Cleaned data saved to 'cleaned_reviews.csv'.

Practical No : 8 - Analyze customer purchasing patterns and build a recommender system based on market basket analysis

```
# Load required libraries
install.packages("arulesViz")
install.packages("arules")
install.packages("recommenderlab")

library(arules)
library(arulesViz)
library(recommenderlab)
```

 # 1. Transactional Data and Association Rule Mining

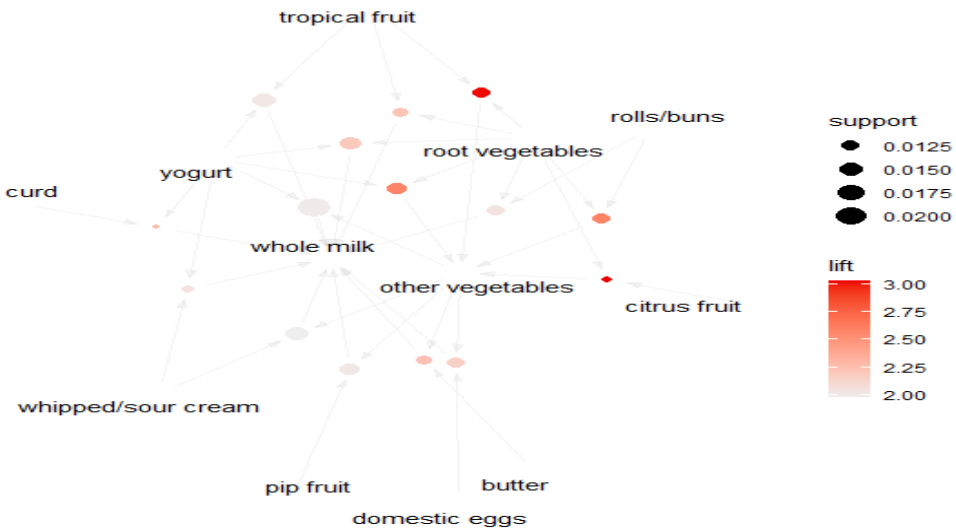
```
# Example dataset
transactions <- read.transactions("https://raw.githubusercontent.com/stedy/Machine-Learning-with-R-datasets/master/groceries.csv",  format = "basket", sep = ",")

# Inspect the data
summary(transactions)
```

```
transactions as itemMatrix in sparse format with
 9835 rows (elements/itemsets/transactions) and
 169 columns (items) and a density of 0.02609146

most frequent items:
      whole milk other vegetables       rolls/buns             soda
            2513             1903             1809             1715
          yogurt          (Other)
            1372            34055

element (itemset/transaction) length distribution:
sizes
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46
  17   18   19   20   21   22   23   24   26   27   28   29   32
  29   14   14    9   11    4    6    1    1    1    1    3    1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   4.409   6.000  32.000

includes extended item information - examples:
            labels
1 abrasive cleaner
2 artif. sweetener
3   baby cosmetics
```

```
# Apply the Apriori algorithm
rules <- apriori(transactions, parameter = list(supp = 0.01, conf = 0.5))
```

Apriori

```
Parameter specification:
confidence minval smax arem  aval originalSupport maxtime support minlen
       0.5    0.1    1 none FALSE            TRUE       5    0.01      1
 maxlen target  ext
    10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 98

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [88 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [15 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].

 # View rules
 inspect(head(sort(rules, by = "lift"), 10))
    lhs                                   rhs               support
[1]  {citrus fruit, root vegetables}   => {other vegetables} 0.01037112
[2]  {root vegetables, tropical fruit} => {other vegetables} 0.01230300
[3]  {rolls/buns, root vegetables}     => {other vegetables} 0.01220132
[4]  {root vegetables, yogurt}         => {other vegetables} 0.01291307
[5]  {curd, yogurt}                    => {whole milk}       0.01006609
[6]  {butter, other vegetables}        => {whole milk}       0.01148958
[7]  {root vegetables, tropical fruit} => {whole milk}       0.01199797
[8]  {root vegetables, yogurt}         => {whole milk}       0.01453991
[9]  {domestic eggs, other vegetables} => {whole milk}       0.01230300
[10] {whipped/sour cream, yogurt}      => {whole milk}       0.01087951
     confidence coverage   lift     count
[1]  0.5862069  0.01769192 3.029608 102
[2]  0.5845411  0.02104728 3.020999 121
[3]  0.5020921  0.02430097 2.594890 120
[4]  0.5000000  0.02582613 2.584078 127
[5]  0.5823529  0.01728521 2.279125  99
[6]  0.5736041  0.02003050 2.244885 113
[7]  0.5700483  0.02104728 2.230969 118
[8]  0.5629921  0.02582613 2.203354 143
[9]  0.5525114  0.02226741 2.162336 121
[10] 0.5245098  0.02074225 2.052747 107

 # Visualize rules
> plot(rules, method = "graph", control = list(type = "items"))
```



 # 2. Building a Recommendation Engine

```
# Load a ratings dataset
data("MovieLense")

# Subset the data for faster computation
ratings <- MovieLense[1:500,]

# Create a recommender system
rec <- Recommender(ratings, method = "UBCF") # User-based collaborative filtering

# Generate recommendations for users
recommendations <- predict(rec, ratings[1:10,], n = 5)

# Convert recommendations to a list
as(recommendations, "list")
```

```
$`0`
[1] "Big Lebowski, The (1998)"
[2] "Tango Lesson, The (1997)"
[3] "Until the End of the World (Bis ans Ende der Welt) (1991)"
[4] "Sense and Sensibility (1995)"
[5] "Winter Guest, The (1997)"

$`1`
[1] "Heavy Metal (1981)"
[2] "Fear of a Black Hat (1993)"
[3] "Forbidden Planet (1956)"
[4] "Paradise Lost: The Child Murders at Robin Hood Hills (1996)"
[5] "Deer Hunter, The (1978)"
```

$`2`
[1] "Heavy Metal (1981)"
[2] "Mystery Science Theater 3000: The Movie (1996)"
[3] "Fear of a Black Hat (1993)"
[4] "Serial Mom (1994)"
[5] "Brady Bunch Movie, The (1995)"
$`3`
[1] "It Happened One Night (1934)" "Jungle Book, The (1994)"
[3] "Deer Hunter, The (1978)" "Man Who Would Be King, The (1975)"
[5] "39 Steps, The (1935)"
$`4`
[1] "Paradise Lost: The Child Murders at Robin Hood Hills (1996)"
[2] "Primal Fear (1996)"
[3] "Wallace & Gromit: The Best of Aardman Animation (1996)"
[4] "Wizard of Oz, The (1939)"
[5] "Clockwork Orange, A (1971)"
$`5`
[1] "Tango Lesson, The (1997)"
[2] "Wallace & Gromit: The Best of Aardman Animation (1996)"
[3] "Forbidden Planet (1956)"
[4] "Ran (1985)"
[5] "Gay Divorcee, The (1934)"
$`6`
[1] "Much Ado About Nothing (1993)" "Philadelphia Story, The (1940)"
[3] "Wings of Desire (1987)" "Innocents, The (1961)"
[5] "Old Man and the Sea, The (1958)"
$`7`
[1] "Scream 2 (1997)" "Wild Things (1998)"
[3] "Lost in Space (1998)" "As Good As It Gets (1997)"
[5] "Usual Suspects, The (1995)"

$`8`
[1] "Nightmare on Elm Street, A (1984)"
[2] "Austin Powers: International Man of Mystery (1997)"
[3] "Lost Highway (1997)"
[4] "Happy Gilmore (1996)"
[5] "Jaws (1975)"

$`9`
[1] "From Dusk Till Dawn (1996)"
[2] "Army of Darkness (1993)"
[3] "Evil Dead II (1987)"
[4] "Bram Stoker's Dracula (1992)"
[5] "Paradise Lost: The Child Murders at Robin Hood Hills (1996)"

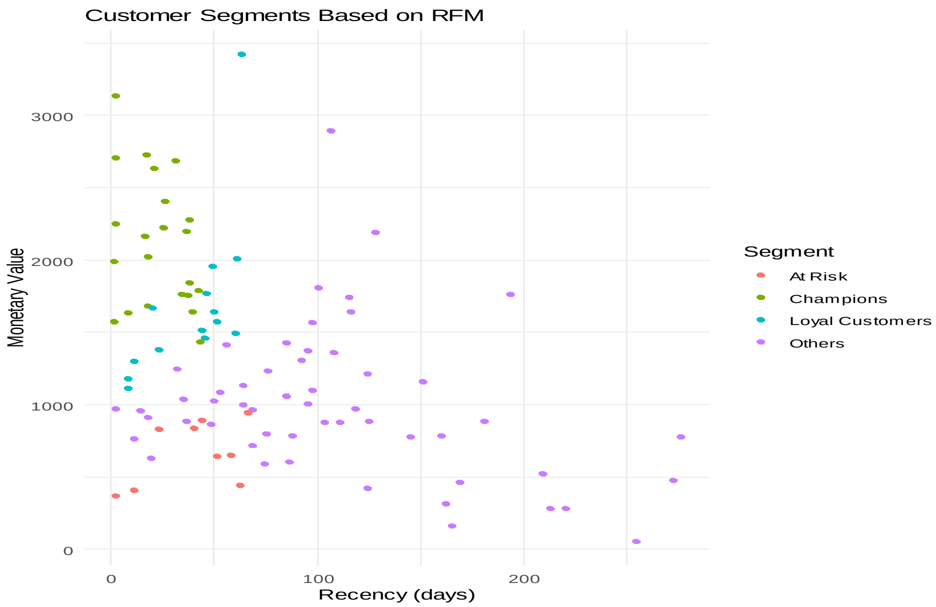
Practical: 9 - Segment customers based on their recency, frequency, and monetary value (RFM) to better target marketing efforts.

```
# Load required libraries
library(dplyr)
library(ggplot2)
library(cluster)

# Sample Transactional Dataset
# Create a dataset with columns: CustomerID, Date, and Amount
set.seed(123)
transactions <- data.frame(
+  CustomerID = sample(1:100, 500, replace = TRUE),
+  Date = sample(seq(as.Date('2023-01-01'), as.Date('2023-12-31'), by = "day"), 500, replace = TRUE),
+  Amount = runif(500, 10, 500)
+ )

# View the dataset
head(transactions)
  CustomerID       Date   Amount
1         31 2023-03-28 272.0319
2         79 2023-03-13 158.0953
3         51 2023-11-11 233.6939
4         14 2023-07-21 198.3278
5         67 2023-06-14 132.0330
6         42 2023-03-22 330.7671
```

 # Step 1: Calculate RFM Metrics

```
# Convert Date column to Date format
transactions$Date <- as.Date(transactions$Date)

# Set the analysis date (e.g., the day the analysis is performed)
analysis_date <- as.Date('2024-01-01')

# Calculate Recency, Frequency, and Monetary metrics
rfm <- transactions %>%
+  group_by(CustomerID) %>%
+  summarise(
+    Recency = as.numeric(analysis_date - max(Date)),  # Days since last purchase
+    Frequency = n(),                    # Number of transactions
+    Monetary = sum(Amount)                   # Total spend
+  )

# View the RFM table
head(rfm)
# A tibble: 6 × 4
  CustomerID Recency Frequency Monetary
       <int   <dbl    <int    <dbl
1        1    276        2     781.
2        2     68        5     962.
3        3    213        1     283.
4        4    151        3    1161.
5        5     62        3     443.
```

```
6        6     38        7    2281.

 # Step 2: Create RFM Scores
 # Divide each metric into quantiles (scored 1-5)
 rfm <- rfm %>%
+   mutate(
+     Recency_Score = ntile(-Recency, 5),  # Negative to assign higher scores for recent purchases
+     Frequency_Score = ntile(Frequency, 5),
+     Monetary_Score = ntile(Monetary, 5)
+   )

 # Calculate the overall RFM score
 rfm <- rfm %>%
+   mutate(RFM_Score = paste0(Recency_Score, Frequency_Score, Monetary_Score))

 # View RFM scores
 head(rfm)
# A tibble: 6 × 8
  CustomerID Recency Frequency Monetary Recency_Score Frequency_Score
       <int   <dbl    <int    <dbl        <int          <int
1        1     276        2     781.          1             1
2        2      68        5     962.          2             3
3        3     213        1     283.          1             1
4        4     151        3    1161.          1             1
5        5      62        3     443.          3             1
6        6      38        7    2281.          4             4
# ℹ 2 more variables: Monetary_Score <int, RFM_Score <chr

 # Step 3: Segment Customers Based on RFM
 # Create customer segments
 rfm <- rfm %>%
+   mutate(
+     Segment = case_when(
+       Recency_Score = 4 & Frequency_Score = 4 & Monetary_Score = 4 ~ "Champions",
+       Recency_Score = 3 & Frequency_Score = 3 & Monetary_Score = 3 ~ "Loyal Customers",
+       Recency_Score = 3 & Frequency_Score <= 2 & Monetary_Score <= 2 ~ "At Risk",
+       TRUE ~ "Others"
+     )
+   )

 # View customer segments
 table(rfm$Segment)

        At Risk      Champions Loyal Customers        Others
            9           22            14           54

 # Step 4: Visualize RFM Segments
 ggplot(rfm, aes(x = Recency, y = Monetary, color = Segment)) +
+   geom_point() +
+   labs(title = "Customer Segments Based on RFM", x = "Recency (days)", y = "Monetary Value") +
+   theme_minimal()
```



 # Step 5: Insights and Marketing Actions

```
 # Summarize segments
 segment_summary <- rfm %>%
+  group_by(Segment) %>%
+  summarise(
+    Avg_Recency = mean(Recency),
+    Avg_Frequency = mean(Frequency),
+    Avg_Monetary = mean(Monetary),
+    Customer_Count = n()
+  )

 # View the segment summary
 segment_summary
# A tibble: 4 × 5
  Segment         Avg_Recency Avg_Frequency Avg_Monetary Customer_Count
  <chr            <dbl        <dbl          <dbl          <int
1 At Risk            39.7         3           669.             9
2 Champions          22.5         7.36       2116.            22
3 Loyal Customers    38.5         6.21       1678.            14
4 Others            108.          4.15       1001.            54
 segment_summary
# A tibble: 4 × 5
  Segment         Avg_Recency Avg_Frequency Avg_Monetary Customer_Count
  <chr            <dbl        <dbl          <dbl          <int
1 At Risk            39.7         3           669.             9
2 Champions          22.5         7.36       2116.            22
3 Loyal Customers    38.5         6.21       1678.            14
4 Others            108.          4.15       1001.            54
```

Practical : 10 - Conduct A/B testing to evaluate the impact of different marketing strategies and make data-driven decisions.

```
# Load required libraries
library(dplyr)
library(ggplot2)
```

 # Step 1: Simulate A/B Testing Data

```
set.seed(123)
n <- 1000  # Total sample size
group <- sample(c("A", "B"), n, replace = TRUE)  # Assign customers to groups randomly
conversion <- ifelse(
  group == "A",
  rbinom(n, 1, 0.12),  # Conversion rate for Group A: 12%
  rbinom(n, 1, 0.15)   # Conversion rate for Group B: 15%
)

# Create the dataset
ab_data <- data.frame(Group = group, Conversion = conversion)

# View a sample of the dataset
head(ab_data)
```

	Group	Conversion
1	A	0
2	A	0
3	A	0
4	B	0
5	A	0
6	B	0

 # Step 2: Summarize Conversion Rates

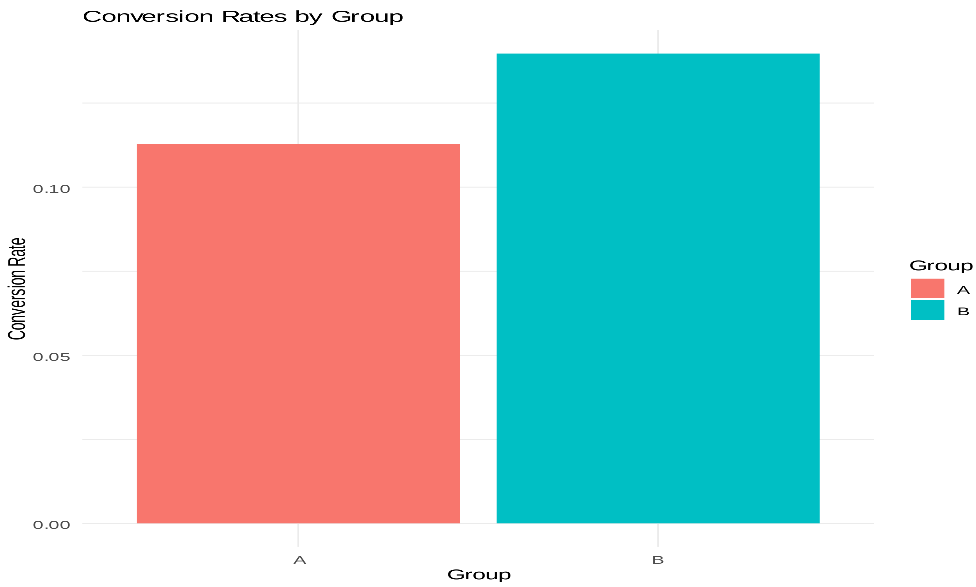
```
summary_table <- ab_data %>%
  group_by(Group) %>%
  summarise(
    Total_Customers = n(),
    Conversions = sum(Conversion),
    Conversion_Rate = mean(Conversion)
  )

# View summary statistics
print(summary_table)
```

A tibble: 2 × 4

	Group	Total_Customers	Conversions	Conversion_Rate
	<chr	<int	<int	<dbl
1	A	506	57	0.113
2	B	494	69	0.140

 # Step 3: Visualize Conversion Rates

```
ggplot(summary_table, aes(x = Group, y = Conversion_Rate, fill = Group))
+  geom_bar(stat = "identity", position = "dodge")
  labs(title = "Conversion Rates by Group", x = "Group", y = "Conversion Rate")
  theme_minimal()
```



 # Step 4: Perform Statistical Test

```
# Null Hypothesis: Conversion rates for Group A and Group B are equal
# Alternative Hypothesis: Conversion rates for Group A and Group B are not equal
ab_test <- prop.test(
  x = summary_table$Conversions,  # Number of conversions in each group
  n = summary_table$Total_Customers,  # Total customers in each group
  alternative = "two.sided"  # Two-tailed test
)

# View test results
print(ab_test)
```

2-sample test for equality of proportions with continuity correction

data: summary_table$Conversions out of summary_table$Total_Customers
X-squared = 1.4218, df = 1, p-value = 0.2331
alternative hypothesis: two.sided
95 percent confidence interval:
-0.07017805 0.01612226

sample estimates:
prop 1 prop 2
0.1126482 0.1396761

 # Step 5: Interpretation

```
if (ab_test$p.value < 0.05) {
  print("Reject the null hypothesis: There is a significant difference between the conversion rates of Group A and Group B.")
} else {
  print("Fail to reject the null hypothesis: No significant difference between the conversion rates of Group A and Group B.")
}
```

[1] "Fail to reject the null hypothesis: No significant difference between the conversion rates of Group A and Group B."