**Output: 1**

> # Load necessary libraries

> library(ggplot2)

> library(dplyr)

> library(corrplot)

> library(summarytools)

> # Load the dataset

> # Replace `mtcars` with your dataset. Use read.csv() or read\_excel() if importing from a file.

> data <- mtcars

> # 1. Basic Information about the Dataset

> cat("Dataset Overview:\n")

Dataset Overview:

> print(dim(data)) # Dimensions of the dataset

[1] **32 11**

> print(str(data)) # Structure of the dataset

'data.frame': 32 obs. of 11 variables:

$ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...

$ cyl : num 6 6 4 6 8 6 8 4 4 6 ...

$ disp: num 160 160 108 258 360 ...

$ hp : num 110 110 93 110 175 105 245 62 95 123 ...

$ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...

$ wt : num 2.62 2.88 2.32 3.21 3.44 ...

$ qsec: num 16.5 17 18.6 19.4 17 ...

$ vs : num 0 0 1 1 0 1 0 1 1 1 ...

$ am : num 1 1 1 0 0 0 0 0 0 0 ...

$ gear: num 4 4 4 3 3 3 3 4 4 4 ...

$ carb: num 4 4 1 1 2 1 4 2 2 4 ...

NULL

> print(summary(data)) # Summary statistics

mpg cyl disp hp drat wt qsec vs am gear carb

Min. 10.4 4 71.1 52.0 2.760 1.513 14.50 0 0 3 1

1st Qu. 15.43 4 120.8 96.5 3.080 2.581 16.89 0 0 3 2

Median 19.2 6 196.3 123 3.7 3.325 17.7 0 0 4 2

Mean 20 6.2 230.7 146.7 3.597 3.217 17.85 0.437 0.406 3.688 2.8

3rd Qu. 22.8 8 326 180.0 3.920 3.610 18.9 1 1 4 4

Max. 33.9 8 472.0 335.0 4.930 5.424 22.90 1 1 5 8

> # 2. Checking for Missing Values

> cat("\nMissing Values Summary:\n")

Missing Values Summary:

> print(sapply(data, function(x) sum(is.na(x)))) # Count missing values in each column

mpg cyl disp hp drat wt qsec vs am gear carb

0 0 0 0 0 0 0 0 0 0 0

> # 3. Detecting Outliers using Boxplots

> cat("\nOutliers Analysis:\n")

Outliers Analysis:

> par(mfrow = c(2, 2)) # Arrange plots in a 2x2 grid

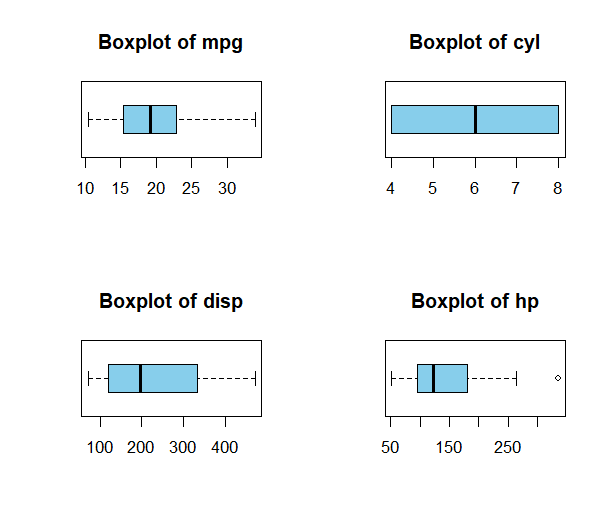
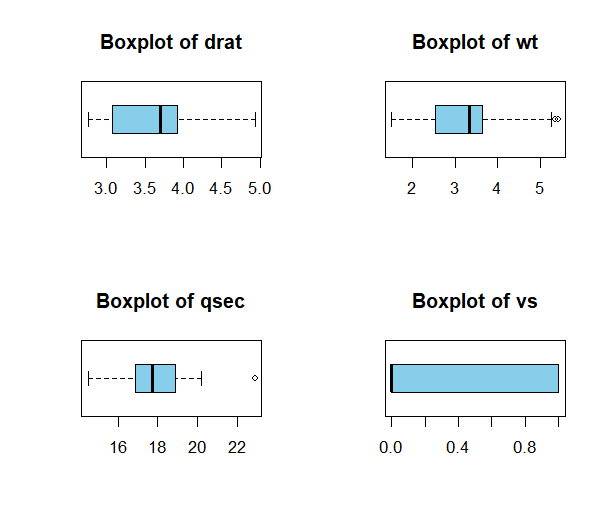
> for (col in names(data)) {

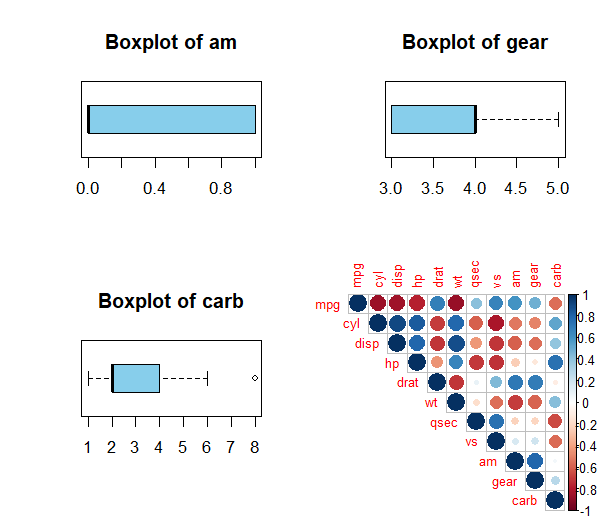
+ if (is.numeric(data[[col]])) {

+ boxplot(data[[col]], main = paste("Boxplot of", col), col = "skyblue", horizontal = TRUE)

+ }

+ }



> # 4. Correlation Matrix for Numeric Variables

> cat("\nCorrelation Matrix:\n")

Correlation Matrix:

> numeric\_data <- select\_if(data, is.numeric) # Select numeric columns

> cor\_matrix <- cor(numeric\_data) # Calculate correlation matrix

> print(cor\_matrix)

mpg cyl disp hp drat wt qsec vs am gear carb

**mpg** 1 -0.85 -0.84 -0.77 0.68 -0.86 0.41 0.66 0.59 0.48 -0.55

**cyl**  -0.85 1 0.9 0.83 -0.69 0.78 -0.59 -0.81 -0.52 -0.49 0.52

**disp**  -0.84 0.9 1 0.79 -0.71 0.88 -0.43 -0.71 -0.59 -0.55 0.39

**hp**  -0.77 0.83 0.79 1 -0.44 0.65 -0.7 -0.72 -0.24 -0.12 0.74

**drat**  0.68 -0.69 -0.71 -0.44 1 -0.71 0.09 0.44 0.71 0.69 -0.09

**wt** -0.86 0.78 0.88 0.65 -0.71 1 -0.17 -0.55 -0.69 -0.58 0.42

**qsec** 0.41 -0.59 -0.43 -0.7 0.09 -0.17 1 0.74 -0.22 -0.21 -0.65

**vs**  0.66 -0.81 -0.71 -0.72 0.44 -0.55 0.744 1 0.16 0.20 -0.56

**am**  0.59 -0.52 -0.59 -0.24 0.71 -0.69 -0.22 0.16 1 0.79 0.05

**gear** 0.48 -0.49 -0.55 -0.12 0.69 -0.58 -0.21 0.20 0.79 1 0.27

**carb**  -0.55 0.52 0.39 0.74 -0.09 0.42 -0.65 -0.56 0.05 0.27 1

> corrplot(cor\_matrix, method = "circle", type = "upper", tl.cex = 0.8)

> # 5. Visualizing Pairwise Relationships (Scatterplot Matrix)

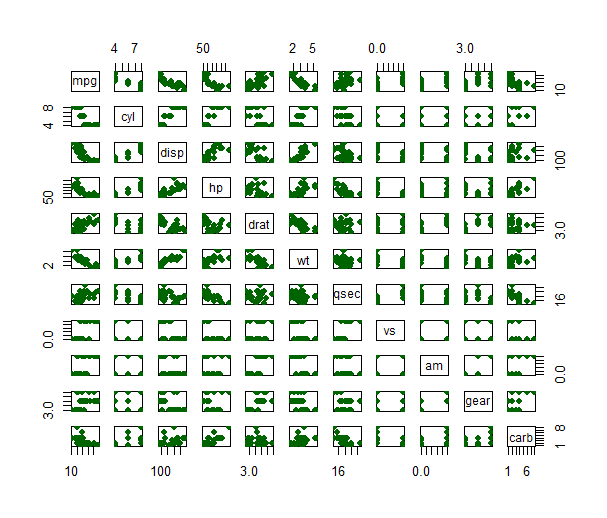
> cat("\nScatterplot Matrix:\n")

Scatterplot Matrix:

> pairs(numeric\_data, col = "darkgreen", pch = 19)

> # 6. Distribution of Each Column

> cat("\nDistribution of Each Column:\n")



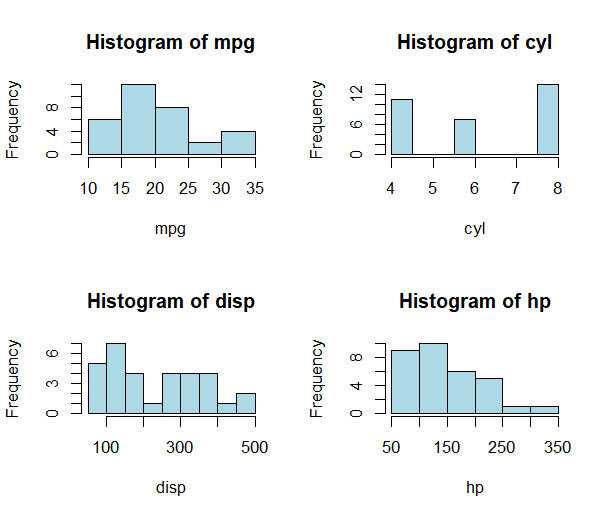
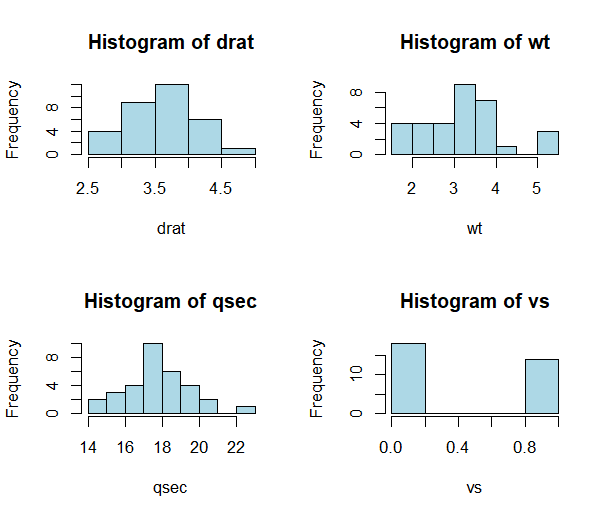
Distribution of Each Column:

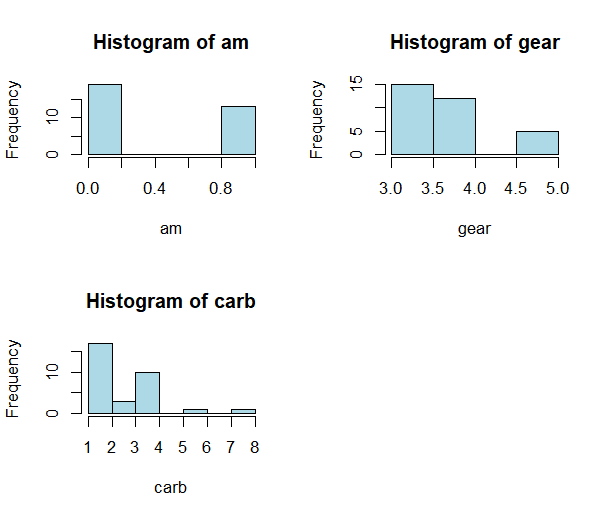
> par(mfrow = c(2, 2)) # Reset plot layout

> for (col in names(numeric\_data)) {

+ hist(numeric\_data[[col]], main = paste("Histogram of", col), col = "lightblue", xlab = col)

+ }



> # 7. Summary Report

> cat("\nDetailed Summary Report:\n")

Detailed Summary Report:

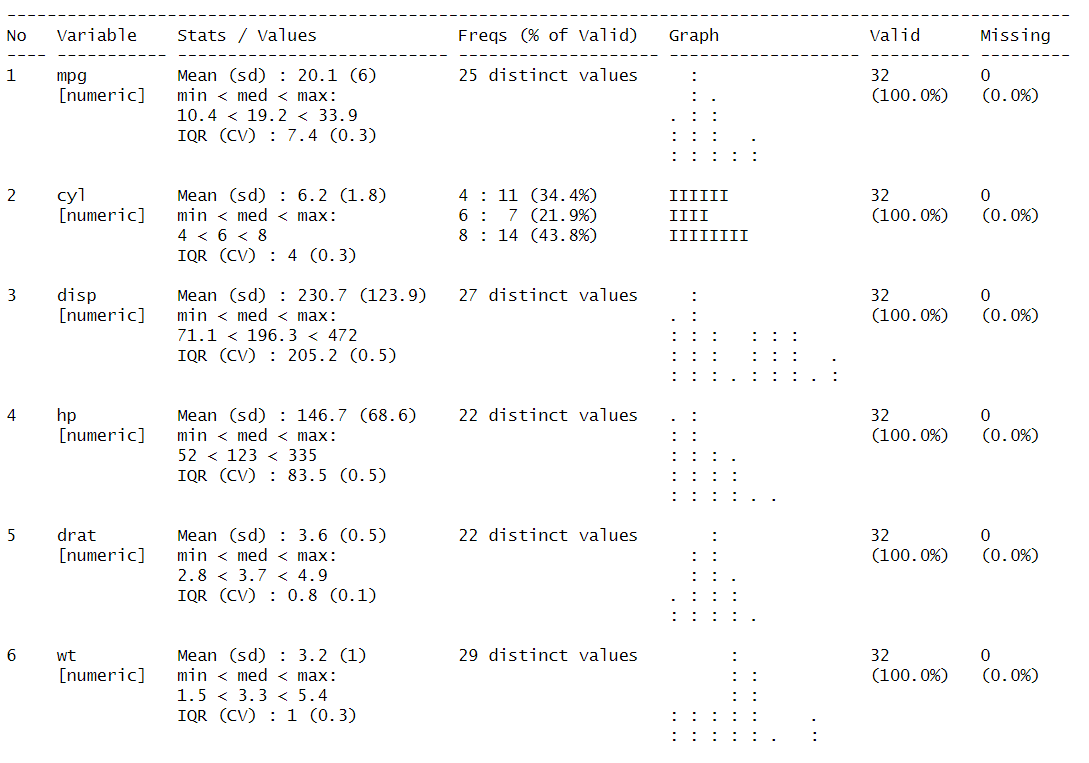
> print(dfSummary(data))

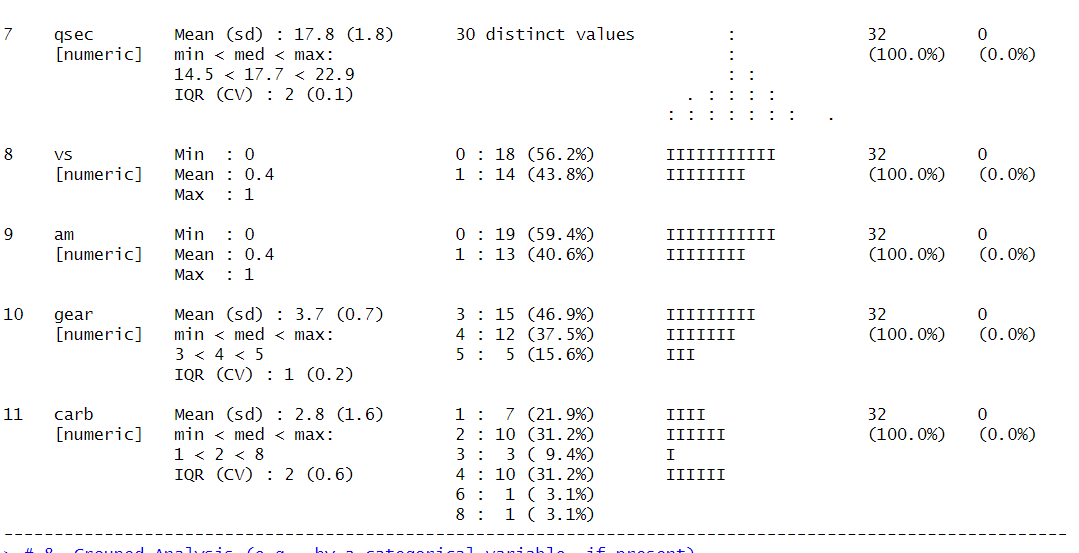
Data Frame Summary

data

Dimensions: 32 x 11

Duplicates: 0





> # 8. Grouped Analysis (e.g., by a categorical variable, if present)

> if ("cyl" %in% names(data)) {

+ cat("\nGrouped Analysis by 'cyl':\n")

+ print(data %>%

+ group\_by(cyl) %>%

+ summarise(across(everything(), mean, na.rm = TRUE)))

+ }

Grouped Analysis by 'cyl':

# A tibble: 3 × 11

cyl mpg disp hp drat wt qsec vs am gear carb

*<dbl>* *<dbl>*  *<dbl>* *<dbl>* *<dbl>* *<dbl>* *<dbl>* *<dbl>* *<dbl>* *<dbl>* *<dbl>*

1 4 26.7 105 82.6 4.07 2.29 19.1 0.909 0.727 4.09 1.55

2 6 19.7 183 122 3.59 3.12 18.0 0.571 0.429 3.86 3.43

3 8 15.1 353 209 3.23 4.00 16.8 0 0.143 3.29 3.5

**Output :2**

> # Load necessary library

> library(ggplot2)

> # Load the dataset

> Mall\_Customers <- read.csv("Mall\_Customers.csv")

> # Data Summary

> summary(Mall\_Customers)

CustomerID Gender Age Income Spending.Score

Min. : 1.00 Length:200 Min. :18.00 Min. : 15.00 Min. : 1.00

1st Qu.: 50.75 Class :character 1st Qu.:28.75 1st Qu.: 41.50 1st Qu.:34.75

Median :100.5 Mode :character Median :36.00 Median : 61.50 Median :50.00

Mean :100.50 Mean :38.85 Mean : 60.56 Mean :50.20

3rd Qu.:150.25 3rd Qu.:49.00 3rd Qu.: 78.00 3rd Qu.:73.00

Max. :200.00 Max. :70.00 Max. :137.00 Max. :99.00

> # Check for missing values

> any(is.na(Mall\_Customers))

**[1] FALSE**

> # Select relevant features for clustering

> data <- Mall\_Customers[, c("Age", "Income", "Spending.Score")]

> # Determine the optimal number of clusters using the Elbow Method

> set.seed(123)

> wss <- sapply(1:10, function(k) {

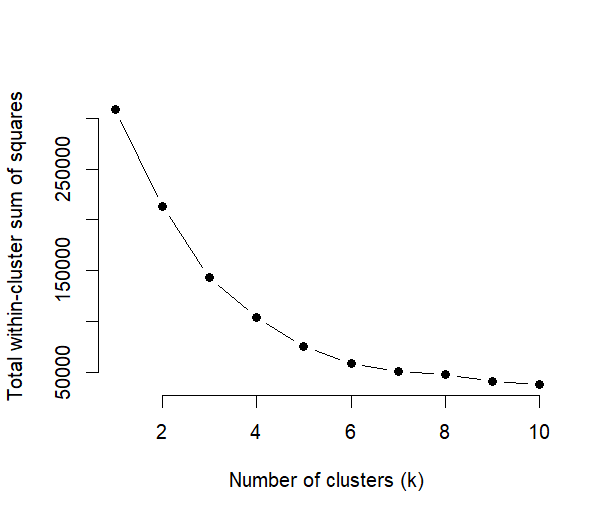
+ kmeans(data, centers = k, nstart = 10)$tot.withinss

+ })

> # Plot the Elbow Method

> plot(1:10, wss, type = "b", pch = 19, frame = FALSE,

+ xlab = "Number of clusters (k)", ylab = "Total within-cluster sum of squares")



> # Apply K-Means Clustering with the optimal number of clusters (e.g., k = 5)

> set.seed(123)

> kmeans\_result <- kmeans(data, centers = 5, nstart = 10)

> # Add cluster labels to the original data

> Mall\_Customers$Cluster <- as.factor(kmeans\_result$cluster)

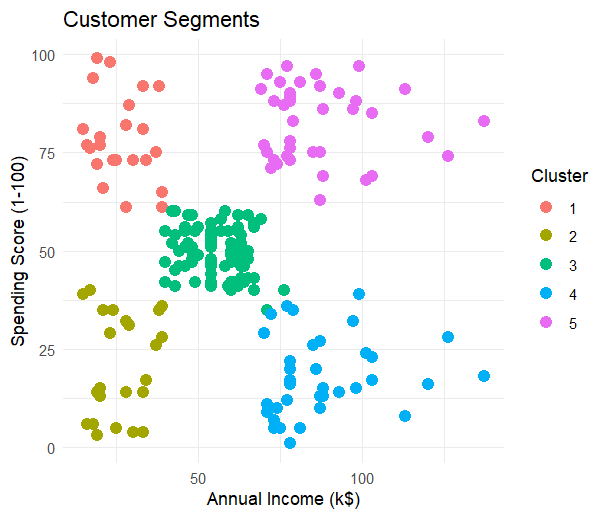
> # Visualize clusters

> ggplot(Mall\_Customers, aes(x = Income, y = Spending.Score, color = Cluster)) +

+ geom\_point(size = 3) +

+ labs(title = "Customer Segments", x = "Annual Income (k$)", y = "Spending Score (1-100)")

+ theme\_minimal()

****

**OutPut :3**

> # Load required 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 1.9.4 ✔ tidyr 1.3.1

✔ purrr 1.0.2

> library(caret)

> # Load the marketing dataset (you can replace it with your own dataset)

> data("marketing", package = "datarium")

> # Inspect the data

> head(marketing)

**Youtube facebook newspaper sales**

1 276.12 45.36 83.04 26.52

2 53.40 47.16 54.12 12.48

3 20.64 55.08 83.16 11.16

4 181.80 49.56 70.20 22.20

5 216.96 12.96 70.08 15.48

6 10.44 58.68 90.00 8.64

> # Split the data into training and test sets

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

> training.samples <- marketing$sales %>% createDataPartition(p = 0.8, list = FALSE)

> train.data <- marketing[training.samples, ]

> test.data <- marketing[-training.samples, ]

> # Build the multiple linear regression model

> model <- lm(sales ~ youtube + facebook + newspaper, data = train.data)

> # Summarize the model

> summary(model)

Call:

lm(formula = sales ~ youtube + facebook + newspaper, data = train.data)

Residuals:

Min 1Q Median 3Q Max

-10.7142 -0.9939 0.3684 1.4494 3.3619

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 3.594142 0.420815 8.541 1.05e-14 \*\*\*

youtube 0.044636 0.001552 28.758 < 2e-16 \*\*\*

facebook 0.188823 0.009529 19.816 < 2e-16 \*\*\*

newspaper 0.002840 0.006442 0.441 0.66

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.043 on 158 degrees of freedom

Multiple R-squared: 0.8955, Adjusted R-squared: 0.8935

F-statistic: 451.2 on 3 and 158 DF, p-value: < 2.2e-16

> # Make predictions on the test set

> predictions <- predict(model, newdata = test.data)

> # Model performance: RMSE (Root Mean Square Error)

> rmse <- RMSE(predictions, test.data$sales)

> print(paste("RMSE: ", round(rmse, 2)))

**[1] "RMSE: 1.97"**

> # R-squared value

> r\_squared <- R2(predictions, test.data$sales)

> print(paste("R-squared: ", round(r\_squared, 2)))

**[1] "R-squared: 0.9"**

> # Example prediction for new data (change the values as needed)

> new\_data <- data.frame(youtube = c(500), facebook = c(200), newspaper = c(100))

> predicted\_sales <- predict(model, newdata = new\_data)

> print(paste("Predicted Sales: ", round(predicted\_sales, 2)))

**[1] "Predicted Sales: 63.96"**

|  |
| --- |
| **Output:4**  # Load the built-in mtcars dataset  > data(mtcars)  > # Perform PCA (scale the data to standardize variables)  > pca\_result <- prcomp(mtcars, scale. = TRUE)  > # Display PCA summary (shows variance explained)  > summary(pca\_result)  Importance of components:  PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8 PC9 PC10 PC11  Standard deviation 2.5707 1.6280 0.79196 0.51923 0.47271 0.46000 0.3678 0.35057 0.2776 0.22811 0.1485  Proportion of Var 0.6008 0.2409 0.05702 0.02451 0.02031 0.01924 0.0123 0.01117 0.0070 0.00473 0.0020  Cumulative Prop 0.6008 0.8417 0.89873 0.92324 0.94356 0.96279 0.9751 0.98626 0.9933 0.99800 1.0000  > # View the principal component loadings  > pca\_result$rotation  PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8 PC9 PC10 PC11  mpg -0.3625 0.01612 -0.2257 -0.02254 -0.1028 -0.10879 0.36772 0.754091 -0.235701 -0.13928 -0.124895  cyl 0.3739 0.04374 -0.17531 -0.00259 -0.0584 0.1685 0.057277 0.23082 -0.054035 0.8464 -0.140695  disp 0.3681 -0.0493 -0.06148 0.256607 -0.3939 -0.33616 0.21430 -0.001142 -0.19842 -0.0493 0.660606  hp 0.3300 0.2487 0.14001 -0.067676 -0.5400 0.07143 -0.00149 0.222358 0.575830 -0.24782 -0.256492  drat -0.2941 0.2746 0.16118 0.85482 -0.07732 0.24449 0.021119 -0.032193 0.046901 0.10149 -0.039530  wt 0.3461 -0.1430 0.34181 0.24589 0.07502 -0.4649 -0.02066 0.008571 -0.35949 -0.0943 -0.567448  qsec -0.2004 -0.46337 0.40316 0.068076 0.16466 -0.33048 0.050010 0.231840 0.528377 0.2706 0.181361  vs -0.3065 -0.23164 0.42881 -0.21484 -0.59953 0.19401 -0.26578 -0.025935 -0.358582 0.15903 0.008414  am -0.2349 0.42941 -0.20576 -0.03046 -0.08978 -0.57081 -0.587305 0.059746 0.0474039 0.17778 0.029823  gear -0.2069 0.46234 0.28977 -0.26469 -0.0483 -0.24356 0.605097 -0.336150 0.001735 0.21385 -0.05350  carb 0.2140 0.41357 0.52854 -0.12678 0.36131 0.18352 -0.17460 0.395629 -0.170640 -0.07225 0.319594  > # Plot the PCA biplot  > biplot(pca\_result, main = "PCA Biplot", scale = 0)    > # Calculate the proportion of variance explained  > variance <- pca\_result$sdev^2  > prop\_variance <- variance / sum(variance)  > # Plot the Scree Plot (Proportion of Variance Explained)  > plot(prop\_variance, type = "b",  + xlab = "Principal Component",  + ylab = "Proportion of Variance Explained",  + main = "Scree Plot")    > # Plot the Cumulative Variance Explained  > plot(cumsum(prop\_variance), type = "b",  + xlab = "Principal Component",  + ylab = "Cumulative Proportion of Variance Explained",  + main = "Cumulative Variance Explained") |
|  |
| |  | | --- | |  | |

Output 5

> # Load required libraries

> install.packages("rpart") # For decision tree

package ‘rpart’ successfully unpacked and MD5 sums checked

Warning in install.packages :

cannot remove prior installation of package ‘rpart’

Warning in install.packages :

problem copying C:\Users\ADMIN\AppData\Local\R\win-library\4.4\00LOCK\rpart\libs\x64\rpart.dll to C:\Users\ADMIN\AppData\Local\R\win-library\4.4\rpart\libs\x64\rpart.dll: Permission denied

Warning in install.packages :

restored ‘rpart’

The downloaded binary packages are in

C:\Users\ADMIN\AppData\Local\Temp\RtmpAfFpMz\downloaded\_packages

> install.packages("rpart.plot") # For visualizing the tree

package ‘rpart.plot’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in

C:\Users\ADMIN\AppData\Local\Temp\RtmpAfFpMz\downloaded\_packages

> # Load required libraries

> install.packages("rpart") # For decision tree

package ‘rpart’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in

C:\Users\ADMIN\AppData\Local\Temp\RtmpAfFpMz\downloaded\_packages

> install.packages("rpart.plot") # For visualizing the tree

package ‘rpart.plot’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in

C:\Users\ADMIN\AppData\Local\Temp\RtmpAfFpMz\downloaded\_packages

> library(rpart)

> library(rpart.plot)

> # Set the working directory to Downloads

> setwd("C:/Users/ADMIN/Downloads")

> # Load the dataset

> churn <- read.csv('WA\_Fn-UseC\_-Telco-Customer-Churn.csv')

> # Check for missing values and remove rows with missing data

> churn <- churn[complete.cases(churn), ]

> # Convert necessary columns to factors

> churn$SeniorCitizen <- as.factor(ifelse(churn$SeniorCitizen == 1, "Yes", "No"))

> churn$Churn <- as.factor(churn$Churn)

> # Simplify tenure into categories

> churn$tenure\_group <- cut(churn$tenure,

+ breaks = c(0, 12, 24, 48, 60, Inf),

+ labels = c("0-12 Month", "12-24 Month", "24-48 Month", "48-60 Month", "> 60 Month"))

> # Remove unnecessary columns

> churn <- churn[, !(names(churn) %in% c("customerID", "tenure"))]

> # Split data into training (70%) and testing (30%) sets

> set.seed(123) # For consistent results

> train\_index <- sample(1:nrow(churn), 0.7 \* nrow(churn))

> train\_data <- churn[train\_index, ]

> test\_data <- churn[-train\_index, ]

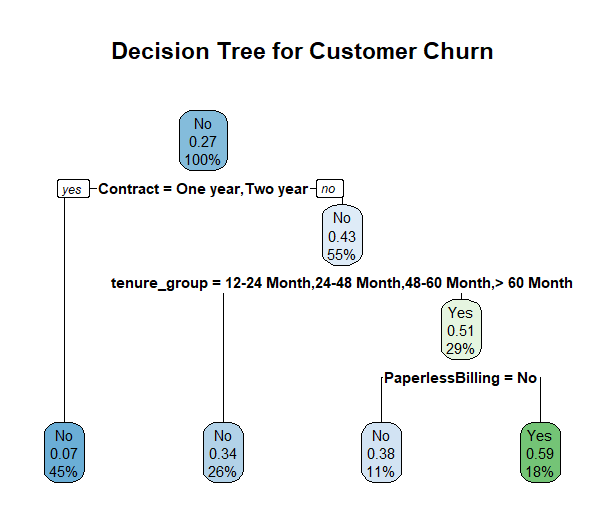
> # Build a decision tree model

> tree\_model <- rpart(Churn ~ Contract + tenure\_group + PaperlessBilling,

+ data = train\_data, method = "class")

> # Visualize the decision tree

> rpart.plot(tree\_model, main = "Decision Tree for Customer Churn")



> # Make predictions on the test set

> predictions <- predict(tree\_model, test\_data, type = "class")

> # Create a confusion matrix

> confusion\_matrix <- table(Predicted = predictions, Actual = test\_data$Churn)

> # Print confusion matrix

> print("Confusion Matrix:")

[1] "Confusion Matrix:"

> print(confusion\_matrix)

Actual

Predicted No Yes

No 1412 344

Yes 138 216

> # Calculate and display accuracy

> accuracy <- sum(diag(confusion\_matrix)) / sum(confusion\_matrix)

> print(paste("Decision Tree Accuracy:", round(accuracy \* 100, 2), "%"))

[1**] "Decision Tree Accuracy: 77.16 %"**