BRAINWARE UNIVERSITY

398, Ramkrishnapur Road, Barasat, North 24 Parganas, Kolkata - 700 125

Laboratory Assignment Submission

Session - 2024 - 25

Name of the Department:-

Programme Name: -

Semester / Year:-

Course Code: -

Course Name: -

Name of the Student:-

Roll No :-

Registration No :-

Student Code : -

INDEX

6	**Assignment – 6** Write an R script to: 1. Create a data frame with the following columns: Name, Age, Marks. 2. Display the structure of the data frame using str(). 3. Use dim(), nrow(), and ncol() to find its dimensions, number of rows, and columns. 4. View the first and last few rows using head() and tail() • Create a data frame with columns: "Name", "Age", and "Score" and display it. • Access the following from the data frame you created: • All rows of the "Name" column. • The first row of the data frame. • The "Age" and "Score" columns for the first two rows. • Apply the following functions to your data frame and interpret the output: • dim() • nrow() • ncol() • str() • summary() • names() • head() • tail()	21-27	27/03/2025			
7	**Assignment – 7** • Add a new column, "Grade", to your data frame with values "A", "B", "A". • Add a new row to your data frame with the details of another student. • Combine two data frames with identical columns using rbind(). • Add a new column, "Hobbies", to your data frame using cbind(). • Merge two data frames using a common column ("ID") and display the result.	28-31	27/03/2025			
8	**Assignment – 8** 1. Perform the following mathematical operations using R: a) Addition, subtraction, multiplication, and division of two numbers.	32-35	03/04/2025			

	b) Calculate the square root, factorial, and exponential of a number. c) Compute the sine, cosine, and tangent of an angle in both degrees and radians. 2.Evaluate the following mathematical expression in R: $\frac{(3x^2+5x+2)}{x^2+1}$, for $x = 1, 2, \dots, 10$. 3. Use R to calculate the following for a given vector of numbers: a) Sum and product of all elements. b) Mean, median, and standard deviation.					
9	**Assignment – 9** Create a box plot for a numerical column in a dataset and identify any outliers.	36	10/04/2025			
10	**Assignment – 10** 1.Use R to generate the frequency distribution of a categorical variable. 2.Create a histogram for a numerical column in a dataset.Analyze and interpret the shape of the histogram (e.g.,skewness,modality).	37-38	17/04/25			
11	**Assignment – 11** 1.Write a dataset to a CSV file and read it back into R. Display the contents of the loaded data. 2. Load an Excel file into R using an appropriate library and display its content. 3.Write an R script to load a CSV file, convert it into a data frame, and display its content. 4.Read a text file containing tab-delimited data into R and convert it into a data frame. 5. Save a subset of a data frame to a new CSV file with a different name.	39-42	24/04/25			
12	**Assignment – 12** 1.Create the following plots using a dataset in R: a) Line plot for numerical data. b) Scatter plot between two variables. c) Bar chart for categorical data. d) Histogram for numerical data. 2. Customize the visualizations created in question 14 by adding: a) Titles, axis labels, and legends. b) Different colors and line types for the plots.	43-49	17/04/25			

13	**Assignment – 13** 1.Create a pie chart to represent the proportion of categories in a dataset. Use the ggplot2 package to create the following: a) A scatter plot with a regression line. b) A grouped bar chart for categorical data. 2.Generate a heatmap for a correlation matrix using a dataset in R.	50-53	17/04/25			
14	**Assignment – 14** 1.Compute the covariance between multiple pairs of numerical variables in a dataset.	54	01/24/25			
15	**Assignment – 15** 1.Create a correlation matrix for a dataset and visualize it using a heatmap. 2.Interpret the strength and direction of relationships between variables based on the correlation values obtained.	55-57	01/05/25			
16	**Assignment – 16** 1.Analyze the regression model output and interpret the following: a) R-squared value. b) Coefficients of the regression equation. c) p-values of the model terms.	58-59	15/05/25			
17	**Assignment – 17** 1.Train a regression model using a training dataset. Use it to predict outcomes for a test dataset. 2.Evaluate the prediction accuracy of the model using the following metrics: a) Mean Absolute Error (MAE). b) Root Mean Squared Error (RMSE).	60-62	15/05/25			
18	**Assignment – 18** 1. Implement a k-Nearest Neighbors (kNN) classification model on a dataset. Evaluate the classification performance. 2. Build a decision tree model using a dataset. Visualize the tree structure. 3. Evaluate the classification models using the following metrics: a) Precision. b) Recall.1 c) F1-Score.	63-67	15/05/25			

19	**Assignment – 19** 1.Use the Boston dataset from the MASS package to predict medv(median home value) based on lstat (percentage of lower status population). Evaluate the residuals and R-squared value.	68-69	22/05/25			
20	**Assignment – 20** Fit a polynomial regression model to predict mpg based on wt and visualize the results with a smooth curve.	70	22/05/25			
21	**Assignment – 21** Use the rpart package to build a decision tree classifier for the iris dataset, predicting the species of flowers.	71	22/05/25			
22	**Assignment – 22** Use the class package to build a KNN classifier for the iris dataset. Experiment with different values of k.	72-74	22/05/25			
23	**Assignment – 23** Compare classification models (e.g., logistic regression, decision tree, SVM) on the iris dataset using metrics such as accuracy, precision, recall, and F1 score.	75-78	22/05/25			

Assignment-1

Create a numeric vector of 10 elements and perform the following operations:

• Calculate the sum & mean of the vector.

• Access the 3rd and 5th elements of the vector.

Code-

```
num_vector <- c(5, 12, 8, 20, 15, 25, 30, 10, 18, 22)
vector_sum <- sum(num_vector)
vector_mean <- mean(num_vector)
third_element <- num_vector[3]
fifth_element <- num_vector[5]
cat("Sum of vector:", vector_sum, "\n")
cat("Mean of vector:", vector_mean, "\n")
cat("3rd Element:", third_element, "\n")
cat("5th Element:", fifth_element, "\n")
```

Ouput-

```
Sum of vector: 165
Mean of vector: 16.5
3rd Element: 8
5th Element: 15
```

Create a list containing:

• Your name, age, and a vector of your five favorite numbers.

• Add a new element to the list (e.g., "Favorite Color").

Code-

```
my_list <- list(
  Name = "Samrat Pal",
  Age = 20,
  Favorite_Numbers = c(10, 11, 19, 28, 35)
)
my_list$Favorite_Color <- "Blue"
print(my_list)
```

Output-

```
$Name
[1] "Samrat Pal"


$Age
[1] 20


$Favorite_Numbers
[1]  10 11 19 28 35


$Favorite_Color
[1] "Blue"
```

Assignment-2

Write an R program to:

Create a 3x3 matrix with numbers from 1 to 9 and perform:

o Transpose of the matrix.

o Row-wise and column-wise sums.

Code-

```
my_matrix <- matrix(1:9, nrow = 3, ncol = 3, byrow = TRUE)
transposed_matrix <- t(my_matrix)
row_sums <- rowSums(my_matrix)
col_sums <- colSums(my_matrix)
cat("Original Matrix:\n")
print(my_matrix)
cat("\nTransposed Matrix:\n")
print(transposed_matrix)
cat("\nRow-wise Sums:\n")
print(row_sums)
cat("\nColumn-wise Sums:\n")
print(col_sums)
```

Output-

```
Original Matrix:
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

```
Transposed Matrix:
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

Row-wise Sums:
[1]  6 15 24

Column-wise Sums:
[1] 12 15 18
```

Create a 2x2x3 array with numbers from 1 to 12, name its rows, columns, and dimensions, and access specific elements.

Code-

```
my_array <- array(1:12, dim = c(2, 2, 3))
dimnames(my_array) <- list(
  Rows = c("Row1", "Row2"),
  Columns = c("Col1", "Col2"),
  Dimensions = c("Matrix1", "Matrix2", "Matrix3")
)
cat("Array with named dimensions:\n")
print(my_array)
element_1 <- my_array["Row1", "Col1", "Matrix1"]  # Access element in first matrix
element_2 <- my_array["Row2", "Col2", "Matrix2"]  # Access element in second matrix
cat("\nAccessed Elements:\n")
cat("Element at (Row1, Col1, Matrix1):", element_1, "\n")
cat("Element at (Row2, Col2, Matrix2):", element_2, "\n")
```

Output-

```
Array with named dimensions:
, , Dimensions = Matrix1

     Col1 Col2
Row1    1    3
Row2    2    4

, , Dimensions = Matrix2

     Col1 Col2
Row1    5    7
Row2    6    8

, , Dimensions = Matrix3

     Col1 Col2
Row1    9   11
Row2   10   12

Accessed Elements:
Element at (Row1, Col1, Matrix1): 1
Element at (Row2, Col2, Matrix2): 8
```

Assignment-3

Write an R program to:

1. Check if a number entered by the user is positive, negative or zero using if-else.

Code-

```
a <- 10
if(a>1){
  print("the number is positive")
}else if(a<1){
  print("the number is negetive")
}else{
  print("the number is one")
}
```

Output-

```
the number is positive
```

2. Use a for loop to print the first 10 natural numbers.

Code-

```
for (i in 1:10) {
  cat(i, "\n")
}
```

Output-

```
1
2
3
4
5
6
7
8
9
10
```

3. Use a while loop to calculate the factorial of a given number.

Code-

```
num <- 5
fact <- 1
i <- 1
while(i<=num){
  fact<-fact*i
  i<-i+1
}
print(fact)
```

Output-

```
[1] 120
```

4. Demonstrate the use of break and next statements in a repeat loop.

Code-

```
i <- 1
repeat{
  if(i%%2==0){
    i <- i+1
    next
  }
  print(i)
  if(i==9){
    break
  }
  i <- i+1
}
```

Output-

```
[1] 1
[1] 3
[1] 5
[1] 7
[1] 9
```

Assignment-4

A.Create a vector of numbers from 1 to 10.

1. Calculate the mean(), sum(), min(), and max() of the vector.

2. Generate a sequence from 5 to 50 with a step of 5 using seq().

3. Concatenate the numbers and print them as a single string using paste().

Code-

```
numbers <- 1:10


mean_value <- mean(numbers)

sum_value <- sum(numbers)

min_value <- min(numbers)

max_value <- max(numbers)


cat("Mean:", mean_value, "\n")

cat("Sum:", sum_value, "\n")

cat("Min:", min_value, "\n")

cat("Max:", max_value, "\n")


sequence <- seq(5, 50, by = 5)

cat("Sequence:", sequence, "\n")


concatenated_string <- paste(numbers, collapse = " ")

cat("Concatenated String:", concatenated_string, "\n")
```

Output-

```
Mean: 5.5

Sum: 55

Min: 1
```

```
Max: 10
Sequence: 5 10 15 20 25 30 35 40 45 50
```

b. Write a script to manipulate the string "Brainware University".

1. Extract the substring "Brainware" using substr().

2. Split the string into individual words using strsplit().

3. Convert the entire string to uppercase and lowercase using toupper() and tolower().

Code-

```
text <- "Brainware University"
substring_text <- substr(text, 1, 9)
split_text <- strsplit(text, " ")
uppercase_text <- toupper(text)
lowercase_text <- tolower(text)
print(paste("Extracted Substring:", substring_text))
print("Split Words:")
print(split_text)
print(paste("Uppercase:", uppercase_text))
print(paste("Lowercase:", lowercase_text))
```

Output-

```
[1] "Extracted Substring: Brainware"
[1] "Split Words:"
[[1]]
[1] "Brainware"   "University"


[1] "Uppercase: BRAINWARE UNIVERSITY"
[1] "Lowercase: brainware university"
```

C. create a vector using seq() from 1 to 20. Write a program to:

1. Repeat the vector three times using rep().

2. Access the first 5 elements of the vector.

3. Demonstrate vector recycling by adding this vector to another vector of length 5.

Code-

```
vec <- seq(1, 20)

repeated_vec <- rep(vec, times = 3)
print("Repeated Vector:")
print(repeated_vec)

first_five <- vec[1:5]
print("First 5 Elements:")
print(first_five)

short_vec <- c(10, 20, 30, 40, 50)
result_vec <- vec + short_vec
print("Vector Recycling Result:")
print(result_vec)
```

Output-

```
Repeated Vector:
1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
First 5 Elements:
1 2 3 4 5
Vector Recycling Result:
 11 22 33 44 55 16 27 38 49 60 21 32 43 54 65 26 37 48 59 70
```

Assignment-5

Write a script to process the text "R Programming is Fun and Challenging".

1. Extract every second word from the sentence.
2. Count the number of occurrences of vowels (a, e, i, o, u) in the string.
3. Replace the word "Challenging" with "Exciting".

Code-

```
text <- "R Programming is Fun and Challenging"
words <- unlist(strsplit(text, " "))
second_word <- words[seq(2, length(words), by = 2)]
extracted_words <- paste(second_words,collopse = " ")
count_vowels <- function(text){
  sum(length(gregexpr("[aeiouAEIOU]",text,perl = TRUE)))
}
updated_text <- gsub("\\behallenging\\b","Exciting",text)
cat("Extracted words:",extracted_words,"\n")
cat("Vowel Count:",vowel_count,"\n")
cat("Extracted words:",updated_text,"\n")
```

Output-

```
Extracted words: Programming   Fun   Challenging
Vowel Count: 9
Extracted words: R Programming is Fun and Challenging
```

• Create a nested list containing:

1. A data frame with student names, marks, and grades.

2. A vector with the total marks for each student.

3. A list of factors indicating the performance category ("Excellent", "Good", "Average").

Code-

```
students_df <- data.frame(
  Name = c("Rohit", "Srijit", "Samrat", "Rounak"),
  Marks = c(95, 78, 60, 88),
  Grade = c("A", "B", "C", "A")
)
total_marks <- students_df$Marks
performance_categories <- factor(
  c("Excellent", "Good", "Average", "Excellent"),
  levels = c("Average", "Good", "Excellent")
)
nested_list <- list(
  Student_Data = students_df,
  Total_Marks = total_marks,
  Performance_Category = performance_categories
)
print(nested_list)
```

Output-

```
$Student_Data
    Name Marks Grade
1  Rohit    95     A
2 Srijit    78     B
3 Samrat    60     C
4 Rounak    88     A
```

```
$Total_Marks
[1] 95 78 60 88

$Performance_Category
[1] Excellent Good      Average   Excellent
Levels: Average Good Excellent
```

• Write a script to:

1. Access and modify the data frame inside the nested list.

2. Add a new entry for a student.

3. Extract students with "Excellent" performance.

Code-

```
students_df <- data.frame(
  Name = c("Rohit", "Srijit", "Samrat", "Rounak"),
  Marks = c(95, 78, 60, 88),
  Grade = c("A", "B", "C", "A")
)
total_marks <- students_df$Marks
performance_categories <- factor(
  c("Excellent", "Good", "Average", "Excellent"),
  levels = c("Average", "Good", "Excellent")
)
nested_list <- list(
  Student_Data = students_df,
  Total_Marks = total_marks,
  Performance_Category = performance_categories
)
nested_list$Student_Data$Marks[students_df$Name == "Charlie"] <- 65
```

```
new_student <- data.frame(Name = "Suvendu", Marks = 90, Grade = "A")
nested_list$Student_Data <- rbind(nested_list$Student_Data, new_student)
nested_list$Total_Marks <- nested_list$Student_Data$Marks
new_performance <- factor("Excellent", levels = c("Average", "Good", "Excellent"))
nested_list$Performance_Category <- c(nested_list$Performance_Category, new_performance)
excellent_students <- nested_list$Student_Data[nested_list$Performance_Category == "Excellent", ]
print(nested_list)
print("Students with Excellent Performance:")
print(excellent_students)
```

Output-

```
$Student_Data
     Name Marks Grade
1   Rohit    95     A
2  Srijit    78     B
3  Samrat    60     C
4  Rounak    88     A
5 Suvendu    90     A

$Total_Marks
[1] 95 78 60 88 90

$Performance_Category
[1] Excellent Good      Average   Excellent Excellent
Levels: Average Good Excellent

[1] "Students with Excellent Performance:"
     Name Marks Grade
1   Rohit    95     A
4  Rounak    88     A
5 Suvendu    90     A
```

Assignment-6

Write an R script to:

1. Create a data frame with the following columns:

Name, Age, Marks.

Code-

```
students <- data.frame(
  Name = c("Samrat", "Srijit", "Rohit"),
  Age = c(20, 22, 21),
  Score = c(85, 90, 88),
  stringsAsFactors = FALSE
)
print(students)
```

Output-

```
  Name Age Score
 1 Samrat  20    85
 2 Srijit  22    90
 3  Rohit  21    88
```

2. Display the structure of the data frame using str().

Code-

```
students <- data.frame(
  Name = c("Samrat", "Srijit", "Rohit"),
  Age = c(20, 22, 21),
  Score = c(85, 90, 88),
  stringsAsFactors = FALSE
)
```

```
str(students)
```

Output-

```
'data.frame':	3 obs. of  3 variables:
 $ Name : chr  "Samrat" "Srijit" "Rohit"
 $ Age  : num  20 22 21
 $ Score: num  85 90 88
```

3. Use dim(), nrow(), and ncol() to find its dimensions, number of rows, and columns.

Code-

```
students <- data.frame(
  Name = c("Samrat", "Srijit", "Rohit"),
  Age = c(20, 22, 21),
  Score = c(85, 90, 88),
  stringsAsFactors = FALSE
)
dim(students)
nrow(students)
ncol(students)
```

Output-

```
[1] 3 3
[1] 3
[1] 3
```

4. View the first and last few rows using head() and tail()

Code-

```
students <- data.frame(
  Name = c("Samrat", "Srijit", "Rohit"),
  Age = c(20, 22, 21),
  Score = c(85, 90, 88),
  stringsAsFactors = FALSE
)
head(students)
tail(students)
```

Output-

```
Name Age Score
1 Samrat  20    85
2 Srijit  22    90
3  Rohit  21    88

 Name Age Score
1 Samrat  20    85
2 Srijit  22    90
3  Rohit  21    88
```

• Create a data frame with columns: "Name", "Age", and "Score" and display it.

Code-

```
df <- data.frame(
  Name = c("Samrat", "Srijit", "Rohit", "Rounak", "Suvendu"),
  Age = c(25, 30, 22, 35, 28),
  Score = c(85, 90, 78, 88, 92)
)
print(df)
```

Output-

```
int(df)
    Name Age Score
1  Samrat  25    85
2  Srijit  30    90
3   Rohit  22    78
4  Rounak  35    88
5 Suvendu  28    92
```

- Access the following from the data frame you created:
  - All rows of the "Name" column.
  - The first row of the data frame.
  - The "Age" and "Score" columns for the first two rows.

Code-

```
df <- data.frame(
  Name = c("Samrat", "Srijit", "Rohit", "Rounak", "Suvendu"),
  Age = c(25, 30, 22, 35, 28),
  Score = c(85, 90, 78, 88, 92)
)
df$Name  # or df[, "Name"]
df[1, ]
df[1:2, c("Age", "Score")]
```

Output-

```
[1] "Samrat"  "Srijit"  "Rohit"   "Rounak"  "Suvendu"
    Name Age Score
```

```
1 Samrat  25    85

  Age Score
1  25    85
2  30    90
```

• Apply the following functions to your data frame and interpret the output:

• dim()

• nrow()

• ncol()

• str()

• summary()

• names()

• head()

• tail()

Code-

```
df <- data.frame(
  Name = c("Samrat", "Srijit", "Rohit", "Rounak", "Suvendu"),
  Age = c(25, 30, 22, 35, 28),
  Score = c(85, 90, 78, 88, 92)
)
dim(df)
nrow(df)
ncol(df)
str(df)
summary(df)
```

```
names(df)
head(df)
tail(df)
```

Output-

```
[1] 5 3

[1] 5

[1] 3

'data.frame':                                   5 obs. of  3 variables:
 $ Name : chr  "Samrat" "Srijit" "Rohit" "Rounak" ...
 $ Age  : num  25 30 22 35 28
 $ Score: num  85 90 78 88 92

     Name              Age        Score
 Length:5          Min.   :22   Min.   :78.0
 Class :character   1st Qu.:25   1st Qu.:85.0
 Mode  :character   Median :28   Median :88.0
                    Mean   :28   Mean   :86.6
                    3rd Qu.:30   3rd Qu.:90.0
                    Max.   :35   Max.   :92.0
[1] "Name"  "Age"   "Score"

     Name Age Score
1  Samrat  25    85
2  Srijit  30    90
3   Rohit  22    78
4  Rounak  35    88
```

```
5 Suvendu  28    92

     Name Age Score
1  Samrat  25    85
2  Srijit  30    90
3   Rohit  22    78
4  Rounak  35    88
5 Suvendu  28    92
```

Assignment-7

• Add a new column, "Grade", to your data frame with values "A", "B", "A".

Code-

```
df <- data.frame(
  Name = c("Samrat", "Srijit", "Rohit", "Rounak", "Suvendu"),
  Age = c(25, 30, 22, 35, 28),
  Score = c(85, 90, 78, 88, 92)
)
df$Grade <- c("A", "B", "A", "B", "A")
print(df)
```

Output-

```
     Name Age Score Grade
1  Samrat  25    85     A
2  Srijit     30    90     B
3  Rohit     22    78     A
4  Rounak  35    88     B
5 Suvendu  28    92     A
```

• Add a new row to your data frame with the details of another student.

Code-

```
df <- data.frame(
  Name = c("Samrat", "Srijit", "Rohit", "Rounak", "Suvendu"),
  Age = c(25, 30, 22, 35, 28),
  Score = c(85, 90, 78, 88, 92)
)
df$Grade <- c("A", "B", "A", "B", "A")
new_student <- data.frame(Name = "Shivam", Age = 27, Score = 80, Grade = "B")
```

```
df <- rbind(df, new_student)
print(df)
```

Output-

```
    Name Age Score Grade
1  Samrat  25    85     A
2  Srijit  30    90     B
3   Rohit  22    78     A
4  Rounak  35    88     B
5 Suvendu  28    92     A
6  Shivam  27    80     B
```

- Combine two data frames with identical columns using rbind().

Code-

```
df1 <- data.frame(
  Name = c("Samrat", "Srijit"),
  Age = c(25, 30),
  Score = c(85, 90),
  Grade = c("A", "B")
)

df2 <- data.frame(
  Name = c("Rohit", "Rounak"),
  Age = c(22, 35),
  Score = c(78, 88),
  Grade = c("A", "B")
)
combined_df <- rbind(df1, df2)
print(combined_df)
```

Output-

```
  Name Age Score Grade
1 Samrat  25    85     A
2 Srijit  30    90     B
3  Rohit  22    78     A
4 Rounak  35    88     B
```

- Add a new column, "Hobbies", to your data frame using cbind().

Code-

```
df <- data.frame(
  Name = c("Samrat", "Srijit", "Rounak", "Rohit"),
  Age = c(25, 30, 22, 35),
  Score = c(85, 90, 78, 88),
  Grade = c("A", "B", "A", "B")
)
hobbies <- c("Reading", "Cycling", "Painting", "Gaming")
df <- cbind(df, Hobbies = hobbies)
print(df)
```

Output-

```
    Name Age Score Grade  Hobbies
1 Samrat  25    85     A  Reading
2 Srijit  30    90     B  Cycling
3 Rounak  22    78     A Painting
4  Rohit  35    88     B   Gaming
```

Merge two data frames using a common column ("ID") and display the result.

Code-

```
df1 <- data.frame(
  ID = c(1, 2, 3),
  Name = c("Samrat", "Srijit", "Rounak"),
  Age = c(25, 30, 22)
)
df2 <- data.frame(
  ID = c(1, 2, 3),
  Score = c(85, 90, 78),
  Grade = c("A", "B", "A")
)
merged_df <- merge(df1, df2, by = "ID")
print(merged_df)
```

Output-

```
ID   Name Age Score Grade
1  1 Samrat  25    85     A
2  2 Srijit  30    90     B
3  3 Rounak  22    78     A
```

Assignment-8

Perform the following mathematical operations using R:

a) Addition, subtraction, multiplication, and division of two numbers.

Code-

```
a <- 10
b <- 5
sum_result <- a + b
print(paste("Addition:", sum_result))
sub_result <- a - b
print(paste("Subtraction:", sub_result))
mul_result <- a * b
print(paste("Multiplication:", mul_result))
div_result <- a / b
print(paste("Division:", div_result))
```

Output-

```
[1] "Addition: 15"
[1] "Subtraction: 5"
[1] "Multiplication: 50"
[1] "Division: 2"
```

b) Calculate the square root, factorial, and exponential of a number.

Code-

```
num <- 5
sqrt_result <- sqrt(num)
factorial_result <- factorial(num)
```

```
exp_result <- exp(num)
cat("Square root of", num, ":", sqrt_result, "\n")
cat("Factorial of", num, ":", factorial_result, "\n")
cat("Exponential of", num, "(e^num):", exp_result, "\n")
```

Output-

```
Square root of 5 : 2.236068
Factorial of 5 : 120
Exponential of 5 (e^num) : 148.4132
```

c) Compute the sine, cosine, and tangent of an angle in both degrees and radians.

Code-

```
angle_deg <- 45
angle_rad <- angle_deg * pi / 180
sin_rad <- sin(angle_rad)
cos_rad <- cos(angle_rad)
tan_rad <- tan(angle_rad)
sin_deg <- sin(angle_deg * pi / 180)
cos_deg <- cos(angle_deg * pi / 180)
tan_deg <- tan(angle_deg * pi / 180)
cat("Angle:", angle_deg, "degrees /", angle_rad, "radians\n")
cat("Sine (rad):", sin_rad, "\n")
cat("Cosine (rad):", cos_rad, "\n")
cat("Tangent (rad):", tan_rad, "\n")
```

Output-

```
Angle: 45 degrees / 0.7853982 radians
Sine (rad): 0.7071068
Cosine (rad): 0.7071068
Tangent (rad): 1
```

2.Evaluate the following mathematical expression in R:

$\frac{(3x^2+5x+2)}{x^2+1}$, for $x = 1, 2, \ldots, 10$.

Code-

```
x_values <- 1:10
results <- (3*x_values^2 + 5*x_values + 2) / (x_values^2 + 1)
print(results)
```

Output-

```
[1] 5.000000 4.800000 4.400000 4.117647 3.923077 3.783784 3.680000 3.600000 3.536585 3.485149
```

3. Use R to calculate the following for a given vector of numbers:

a) Sum and product of all elements.

Code-

```
numbers <- c(3, 7, 2, 9)
sum_result <- sum(numbers)
product_result <- prod(numbers)
cat("Sum of elements:", sum_result, "\n")
cat("Product of elements:", product_result, "\n")
```

Output-

```
Sum of elements: 21
Product of elements: 378
```

b) Mean, median, and standard deviation.

Code-

```
numbers <- c(10, 20, 30, 40, 50)
mean_result <- mean(numbers)
median_result <- median(numbers)
sd_result <- sd(numbers)
cat("Mean:", mean_result, "\n")
cat("Median:", median_result, "\n")
cat("Standard Deviation:", sd_result, "\n")
```

Output-

```
Mean: 30
Median: 30
Standard Deviation: 15.81139
```

Assignment-9

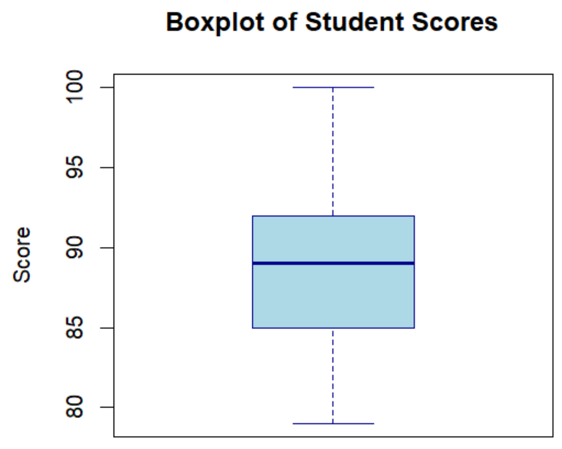
Create a box plot for a numerical column in a dataset and identify any outliers.

Code-

```
student_data <- data.frame(
  Name = c("Samrat", "Srijit", "Rohit", "Rouank", "Suvendu", "Akash"),
  Age = c(22, 24, 23, 21, 25, 30),
  Score = c(88, 92, 79, 85, 90, 100)
)
boxplot(student_data$Score,
        main = "Boxplot of Student Scores",
        ylab = "Score",
        col = "lightblue",
        border = "darkblue")
outliers <- boxplot.stats(student_data$Score)$out
print(outliers)
```

Output-

```
numeric(0)
```



Assignment-10

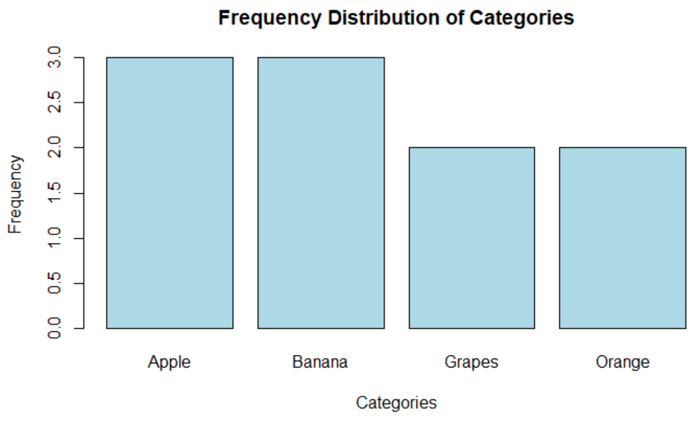
1.Use R to generate the frequency distribution of a categorical variable.

Code-

```
categories <- c("Apple", "Banana", "Apple", "Orange", "Banana", "Apple", "Grapes", "Orange", "Banana", "Grapes")
frequency_distribution <- table(categories)
print(frequency_distribution)
frequency_df <- as.data.frame(frequency_distribution)
colnames(frequency_df) <- c("Category", "Frequency")
print(frequency_df)
barplot(frequency_distribution,
        main = "Frequency Distribution of Categories",
        xlab = "Categories",
        ylab = "Frequency",
        col = "lightblue")
```

Output-

```
categories
 Apple Banana Grapes Orange
     3      3      2      2
  Category Frequency
1    Apple         3
2   Banana         3
3   Grapes         2
4   Orange         2
```

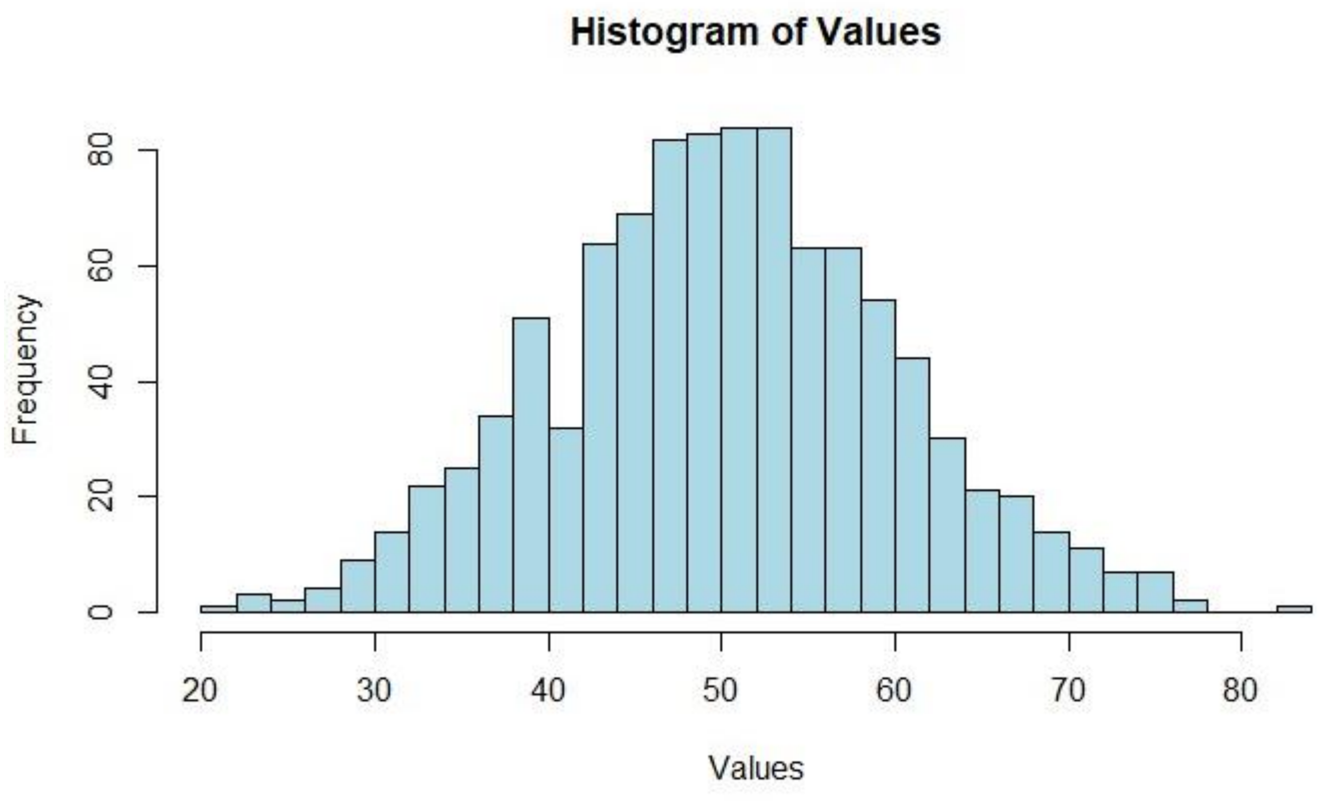


2.Create a histogram for a numerical column in a dataset.Analyze and interpretthe shape of the histogram(e.g.,skewness,modality).

Code-

```
set.seed(123)
data <- data.frame(values = rnorm(1000, mean = 50, sd = 10))
hist(data$values,
    main = "Histogram of Values",
    xlab = "Values",
    ylab = "Frequency",
    col = "lightblue",
    border = "black",
    breaks = 30)
```

Output-



Assignment-11

1.Write a dataset to a CSV file and read it back into R. Display the contents of the loaded data.

Code-

```
data <- data.frame(
  Name = c("Samrat", "Srijit", "Rounak"),
  Age = c(25, 30, 35),
  City = c("New York", "Los Angeles", "Chicago")
)
print(data)
write.csv(data, "C:/Users/Samrat Pal/Downloads/sample_data.csv", row.names = FALSE)
loaded_data <- read.csv("C:/Users/Samrat Pal/Downloads/sample_data.csv")
print(loaded_data)
```

Output-

```
    Name Age        City
1 Samrat  25    New York
2 Srijit  30 Los Angeles
3 Rounak  35     Chicago

    Name Age        City
1 Samrat  25    New York
2 Srijit  30 Los Angeles
3 Rounak  35     Chicago
```

2. Load an Excel file into R using an appropriate library and display its content.

Code-

```
library(readxl)
```

```
excel_data <- read_excel("C:/Users/Samrat Pal/Downloads/sample_data.xlsx")
print(excel_data)
```

Output-

```
  Name     Age City


1 Samrat    25 New York
2 Srijit    30 Los Angeles
3 Rounak    35 Chicago
```

3.Write an R script to load a CSV file, convert it into a data frame, and display its content.

Code-

```
library(readr)
csv_file_path <- "C:/Users/Samrat Pal/Downloads/sample_data.csv"
data_frame <- read_csv(csv_file_path)
print(data_frame)
```

Output-

```
Rows: 3 Columns: 3
── Column specification
Delimiter: ","
chr (2): Name, City
dbl (1): Age


ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
  Name     Age City
1 Samrat    25 New York
2 Srijit    30 Los Angeles
3 Rounak    35 Chicago
```

4.Read a text file containing tab-delimited data into R and convert it into a data frame.

Code-

```
file_path <- "C:/Users/Samrat Pal/Downloads/sample_data.csv"
if (file.exists(file_path)) {
  data <- read.delim(file_path, header = TRUE, sep = "\t")
  cat("File successfully read!\n")
  str(data)
  head(data)
} else {
  cat("Error: File does not exist at the specified path.\n")
}
```

Output-

```
File successfully read!
'data.frame':	3 obs. of  1 variable:
 $ Name.Age.City: chr  "Samrat,25,New York" "Srijit,30,Los Angeles" "Rounak,35,Chicago"
          Name.Age.City
1    Samrat,25,New York
2 Srijit,30,Los Angeles
3     Rounak,35,Chicago
```

5. Save a subset of a data frame to a new CSV file with a different name.

Code-

```
data <- mtcars  # using the built-in mtcars dataset
subset_data <- data[data$mpg > 20, ]
write.csv(subset_data, file = "subset_mtcars.csv", row.names = FALSE)
```

Output-

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

Assignment-12

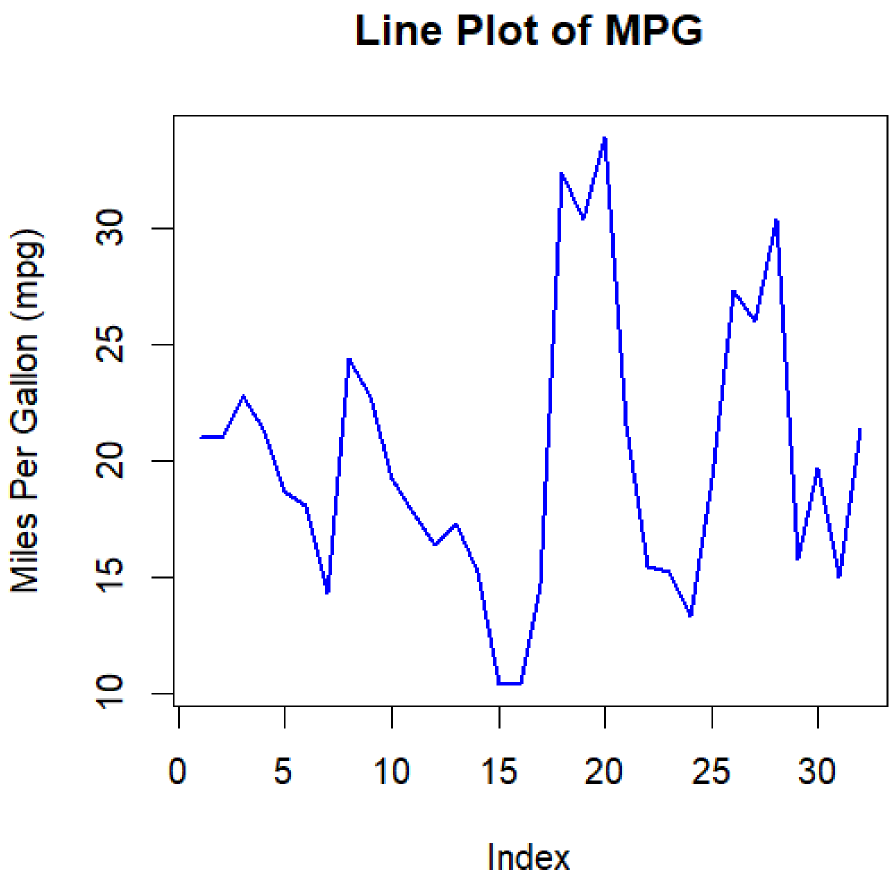
1.Create the following plots using a dataset in R:

a) Line plot for numerical data.

Code-

```
data(mtcars)
plot(mtcars$mpg, type = "l",
    col = "blue",
    lwd = 2,
    xlab = "Index",
    ylab = "Miles Per Gallon (mpg)",
    main = "Line Plot of MPG")
```

Output-

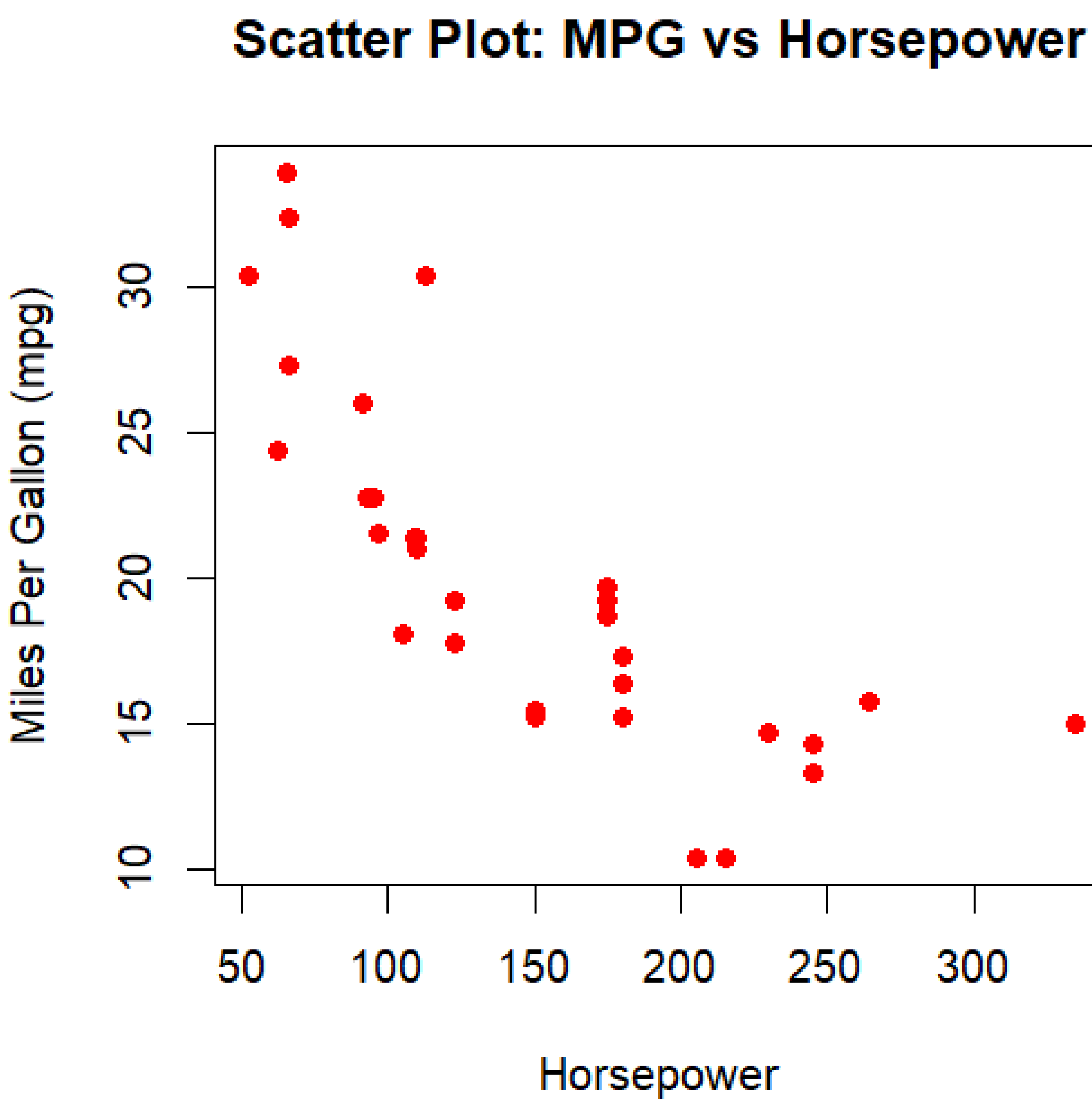


b) Scatter plot between two variables.

Code-

```
data(mtcars)
plot(mtcars$hp, mtcars$mpg,
    col = "red",
    pch = 19,
    xlab = "Horsepower",
    ylab = "Miles Per Gallon (mpg)",
    main = "Scatter Plot: MPG vs Horsepower")
```

Output-

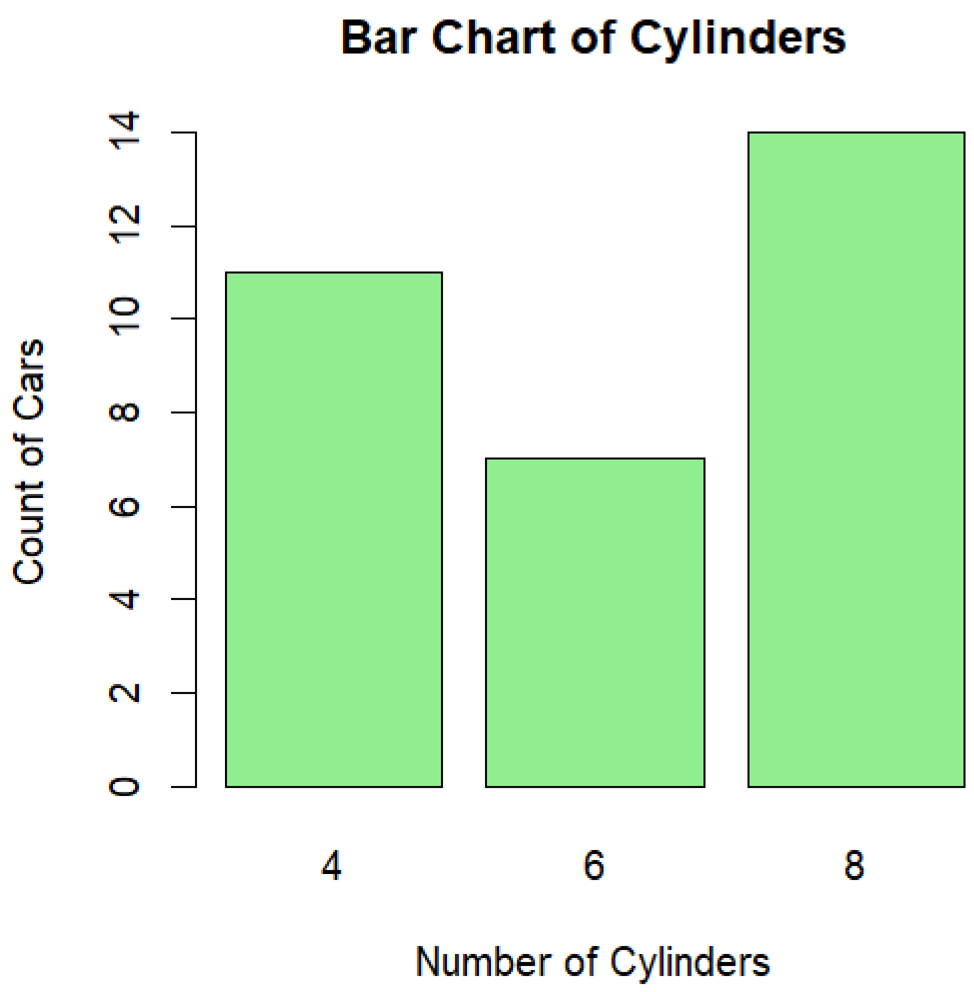


c) Bar chart for categorical data.

Code-

```
data(mtcars)
cyl_factor <- as.factor(mtcars$cyl)
barplot(table(cyl_factor),
        col = "lightgreen",
        xlab = "Number of Cylinders",
        ylab = "Count of Cars",
        main = "Bar Chart of Cylinders")
```

Output-

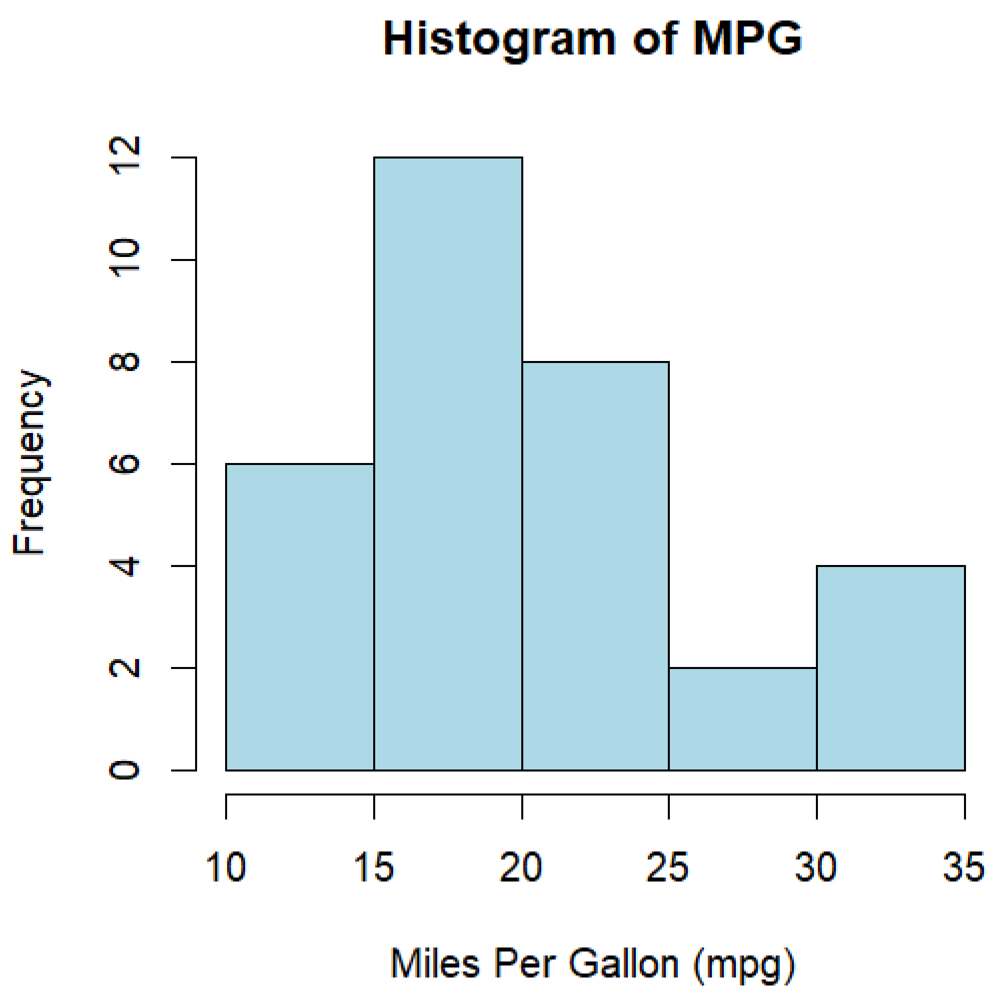


d) Histogram for numerical data.

Code-

```
data(mtcars)
hist(mtcars$mpg,
    col = "lightblue",
    border = "black",
    xlab = "Miles Per Gallon (mpg)",
    main = "Histogram of MPG")
```

Output-



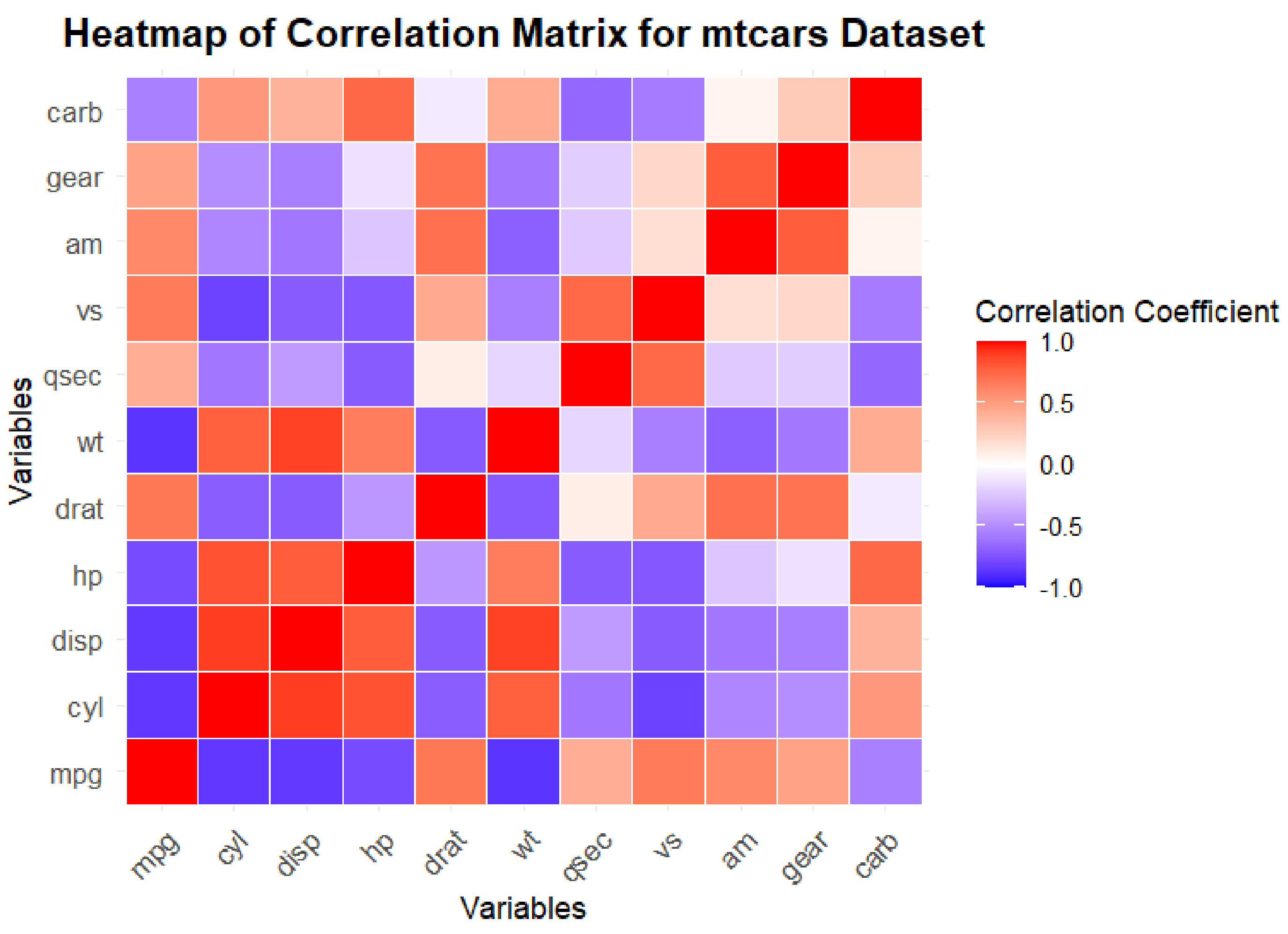
2. Customize the visualizations created in question 14 by adding:

a) Titles, axis labels, and legends.

Code-

```
library(ggplot2)
library(reshape2)
data(mtcars)
cor_matrix <- cor(mtcars)
cor_melted <- melt(cor_matrix)
ggplot(data = cor_melted, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1, 1),
                       name = "Correlation Coefficient") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 10),
        axis.text.y = element_text(size = 10),
        plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
        legend.position = "right") +
  labs(title = "Heatmap of Correlation Matrix for mtcars Dataset",
       x = "Variables",
       y = "Variables")
```

Output-

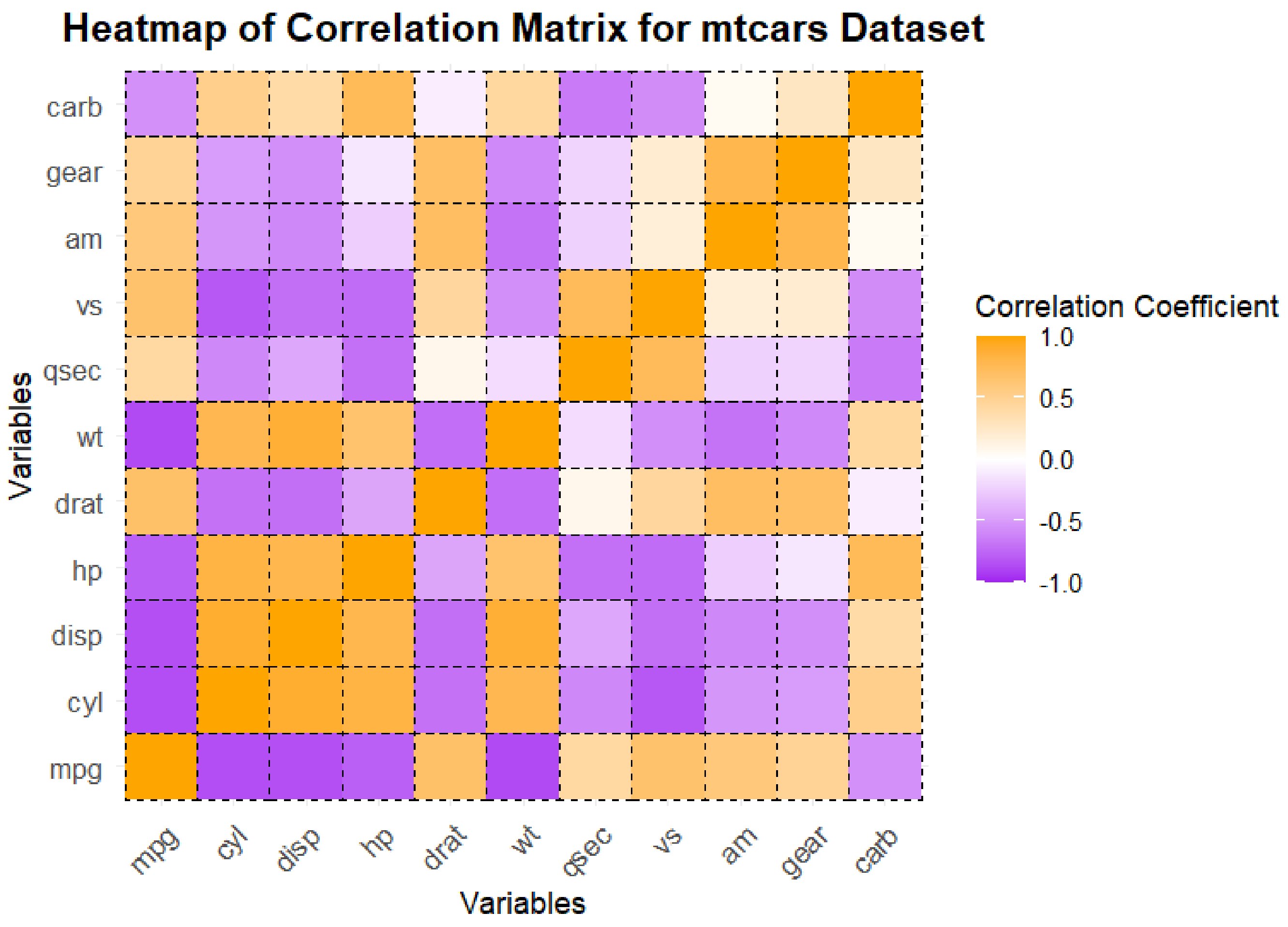


b) Different colors and line types for the plots.

Code-

```
library(ggplot2)
library(reshape2)
data(mtcars)
cor_matrix <- cor(mtcars)
cor_melted <- melt(cor_matrix)
ggplot(data = cor_melted, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile(color = "black", linetype = "dashed") +
  scale_fill_gradient2(low = "purple", mid = "white", high = "orange",
                      midpoint = 0, limit = c(-1, 1),
                      name = "Correlation Coefficient") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 10),
        axis.text.y = element_text(size = 10),
        plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
        legend.position = "right") +
  labs(title = "Heatmap of Correlation Matrix for mtcars Dataset",
       x = "Variables",
       y = "Variables")
```

Output-



Assignment-13

1.Create a pie chart to represent the proportion of categories in a dataset.

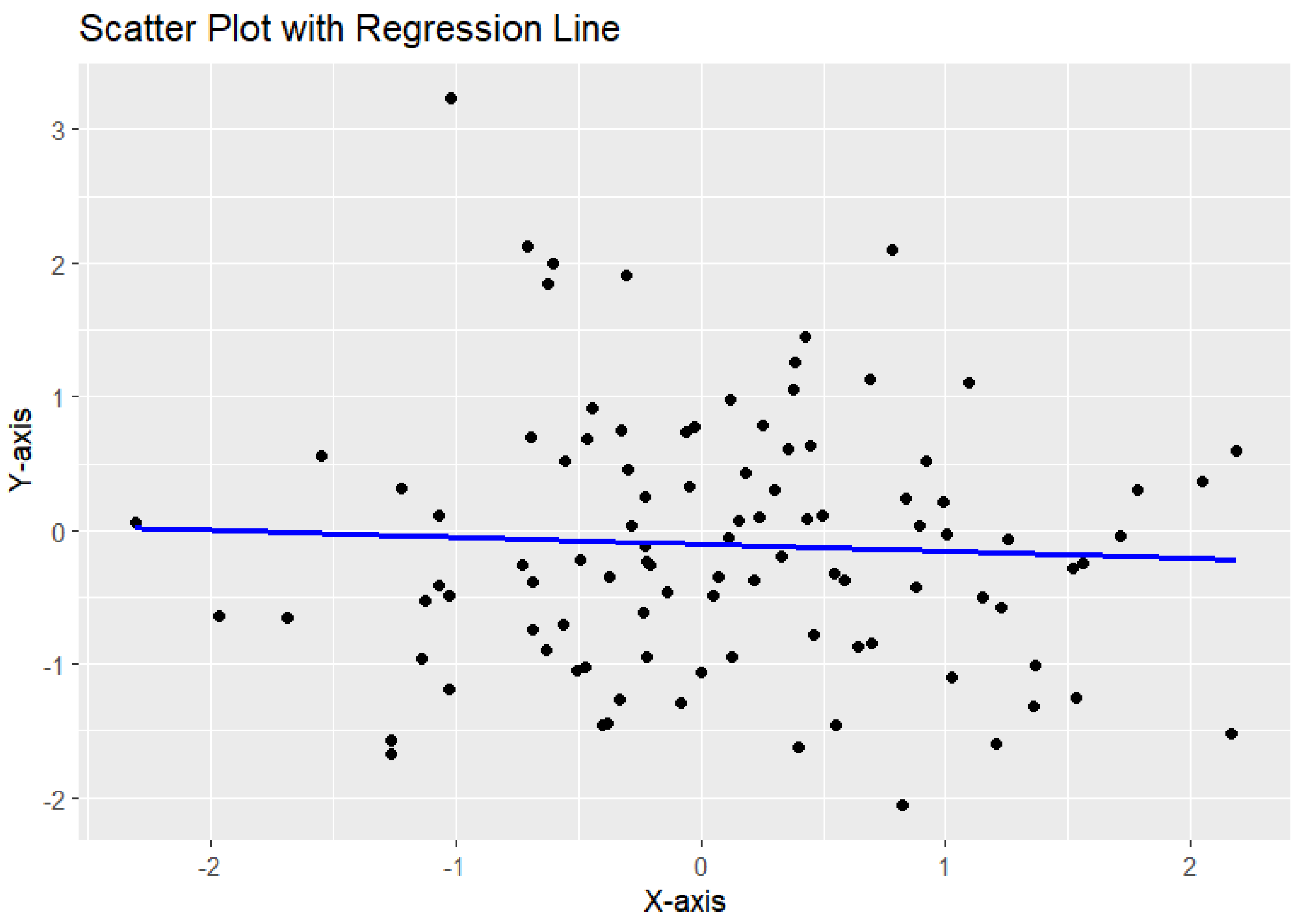
Use the ggplot2 package to create the following:

a) A scatter plot with a regression line.

Code-

```
install.packages("ggplot2”)
install.packages("dplyr”)
library(ggplot2)
library(dplyr)
data <- data.frame(
  category = c("A", "B", "C", "D"),
  values = c(10, 20, 30, 40)
)
pie_chart <- ggplot(data, aes(x = "", y = values, fill = category)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y") +
  theme_void() +
  labs(title = "Proportion of Categories")
print(pie_chart)
set.seed(123)
scatter_data <- data.frame(
  x = rnorm(100),
  y = rnorm(100)
)
scatter_plot <- ggplot(scatter_data, aes(x = x, y = y)) +
  geom_point() +  # Scatter points
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  labs(title = "Scatter Plot with Regression Line", x = "X-axis", y = "Y-axis")
print(scatter_plot)
```

Output-

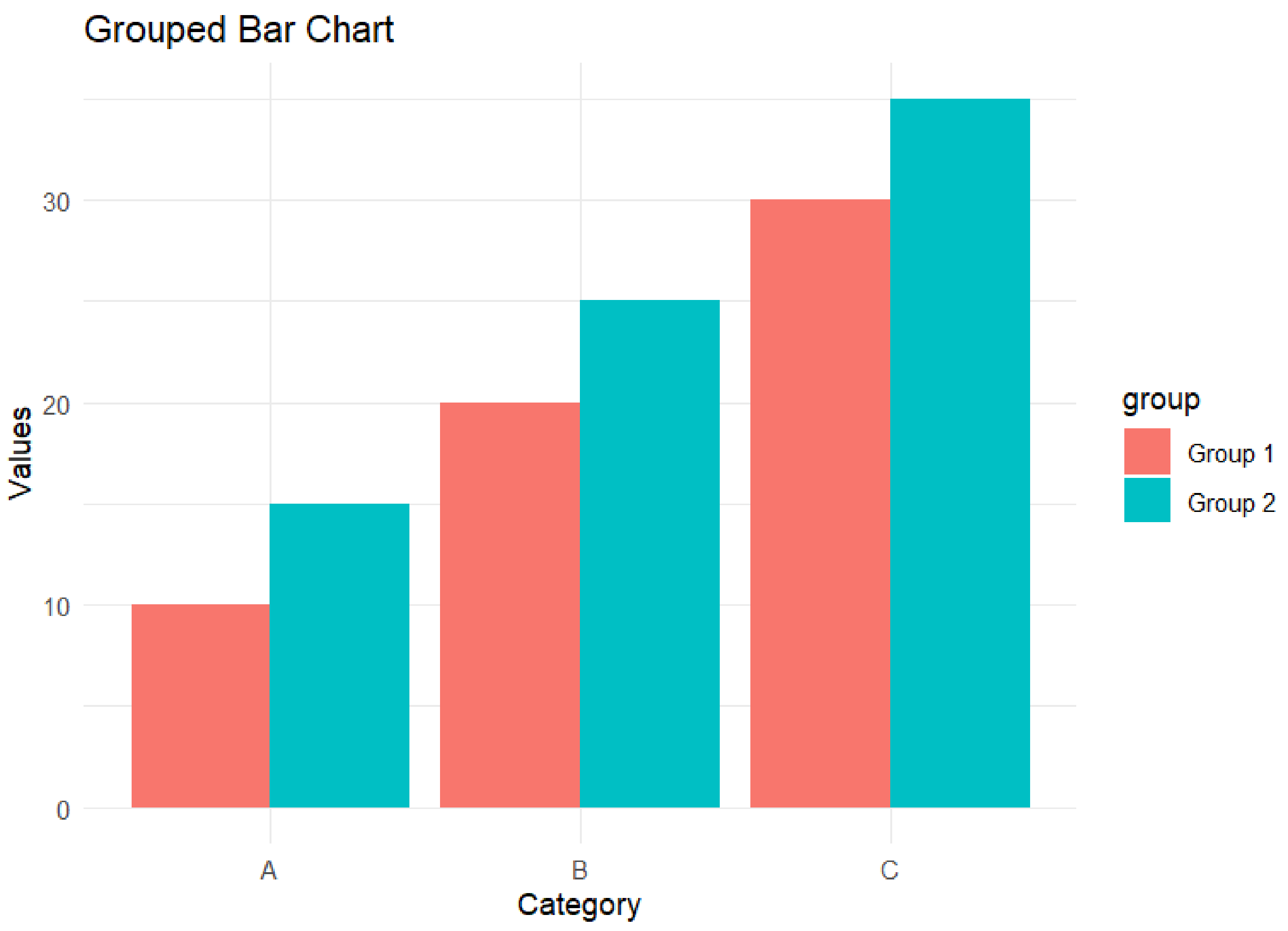


b) A grouped bar chart for categorical data.

Code-

```
library(ggplot2)
library(dplyr)
data <- data.frame(
  category = rep(c("A", "B", "C"), each = 2),
  group = rep(c("Group 1", "Group 2"), times = 3),
  values = c(10, 15, 20, 25, 30, 35)
)
grouped_bar_chart <- ggplot(data, aes(x = category, y = values, fill = group)) +
  geom_bar(stat = "identity", position = "dodge") +  # Use position = "dodge" for grouped bars
  labs(title = "Grouped Bar Chart", x = "Category", y = "Values") +
  theme_minimal()
print(grouped_bar_chart)
```

Output-



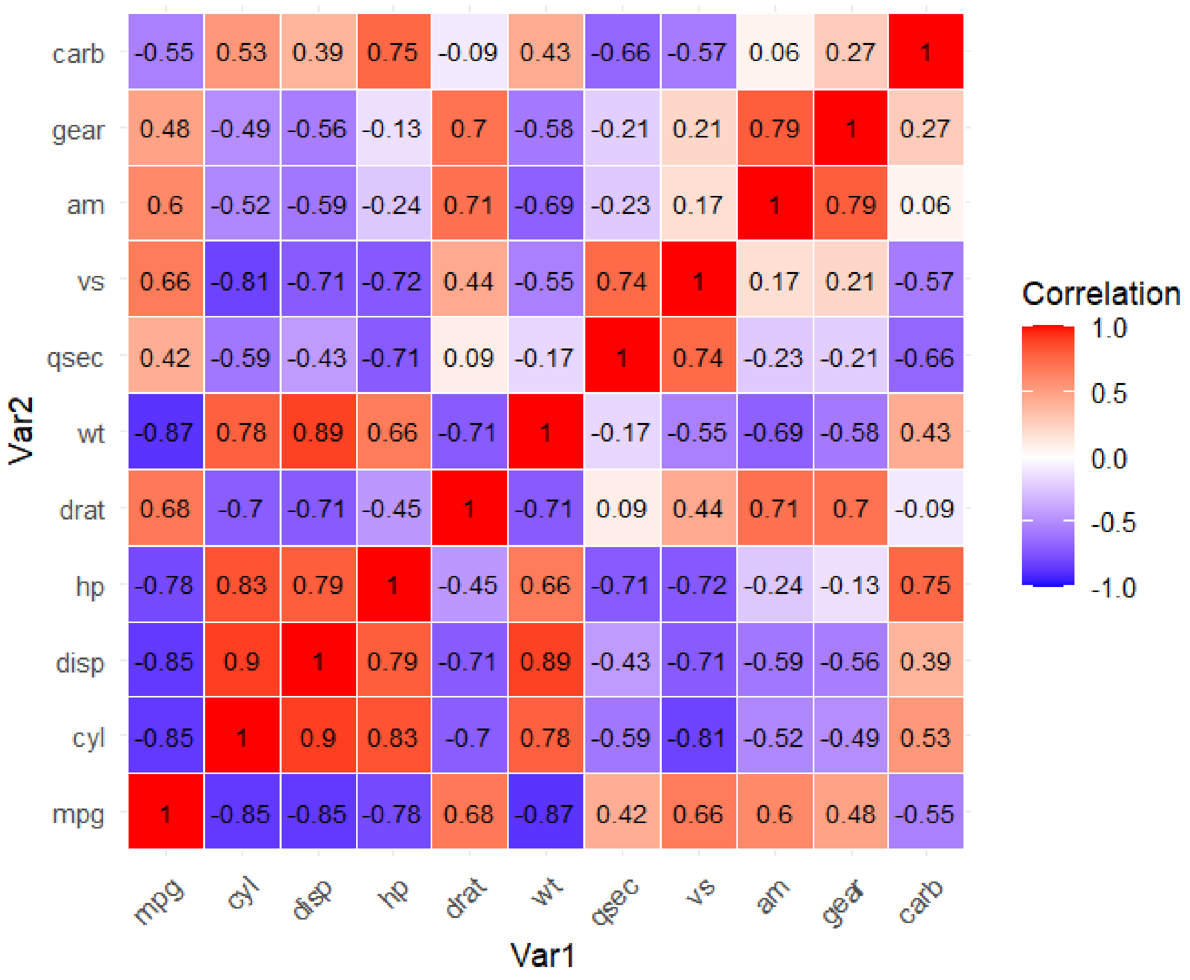
2.Generate a heatmap for a correlation matrix using a dataset in R.

Code-

```
library(ggplot2)
library(reshape2)
library(RColorBrewer)
data <- mtcars
cor_matrix <- round(cor(data), 2)
melted_cor <- melt(cor_matrix)
```

```
ggplot(data = melted_cor, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Correlation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1)) +
  coord_fixed() +
  geom_text(aes(label = value), color = "black", size = 3)
```

Output-



Assignment-14

1.Compute the covariance between multiple pairs of numerical variables in a dataset.

Code-

```
df <- data.frame(
  height = c(150, 160, 170, 180, 190),
  weight = c(65, 72, 78, 85, 90),
  age    = c(25, 30, 35, 40, 45),
  gender = c("F", "M", "M", "F", "M")
)
numeric_df <- df[sapply(df, is.numeric)]
cov_matrix <- cov(numeric_df)
print(cov_matrix)
```

Output-

```
       height weight    age
height  250.0 157.50 125.00
weight 157.5  99.50  78.75
age     125.0  78.75  62.50
```

2.Calculate the correlation coefficient between two numerical columns in a dataset.

Code-

```
df <- data.frame(
  height = c(150, 160, 170, 180, 190),
  weight = c(65, 72, 78, 85, 90)
)
correlation <- cor(df$height, df$weight)
print(correlation)
```

Output-

```
 0.9953501
```

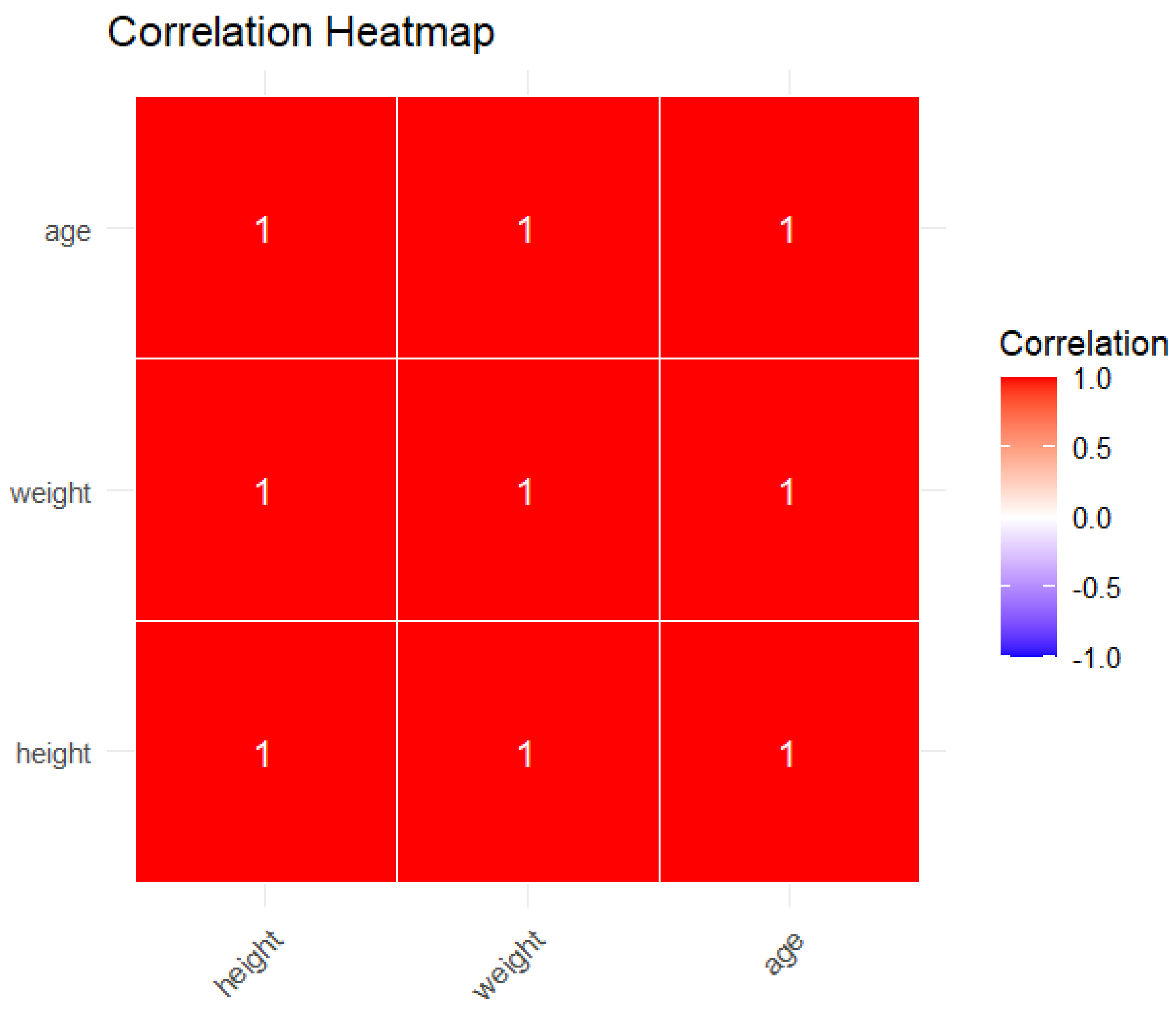
Assignment-15

1.Create a correlation matrix for a dataset and visualize it using a heatmap.

Code-

```
df <- data.frame(
  height = c(150, 160, 170, 180, 190),
  weight = c(65, 72, 78, 85, 90),
  age = c(25, 30, 35, 40, 45)
)
numeric_df <- df[sapply(df, is.numeric)]
cor_matrix <- cor(numeric_df)
corrplot(cor_matrix,
         method = "color",
         addCoef.col = "black",
         tl.col = "black",
         tl.cex = 1.2,
         number.cex = 1.1,
         col = colorRampPalette(c("blue", "white", "red"))(200))
melted_cor <- melt(cor_matrix)
ggplot(data = melted_cor, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile(color = "white") +
  geom_text(aes(label = round(value, 2)), color = "white", size = 4) +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1, 1), space = "Lab",
                       name = "Correlation") +
  theme_minimal() +
  coord_fixed() +
  labs(title = "Correlation Heatmap", x = "", y = "") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
```

Output-



Correlation Heatmap
age
weight
height
1
1
1
1
1
1
1
1
1
height
weight
age
Correlation
1.0
0.5
0.0
-0.5
-1.0

2.Interpret the strength and direction of relationships between variables based on the correlation values obtained.

Answer-

Sure! Here's a quick guide to interpreting the **strength** and **direction** of relationships based on **correlation values**:

1. Direction

- **Positive correlation (r > 0)**: As one variable increases, the other also increases.
- **Negative correlation (r < 0)**: As one variable increases, the other decreases.

2. Strength

(These are general rules; exact interpretation can depend on the field you're working in.)

Correlation Coefficient (r)	**Strength**
0.90 to 1.00 or -0.90 to -1.00	Very strong
0.70 to 0.89 or -0.70 to -0.89	Strong
0.40 to 0.69 or -0.40 to -0.69	Moderate
0.10 to 0.39 or -0.10 to -0.39	Weak
0.00 to 0.09 or -0.00 to -0.09	Very weak or none

Example interpretations:

- **r = 0.82** → Strong positive relationship
- **r = -0.52** → Moderate negative relationship
- **r = 0.15** → Weak positive relationship
- **r = -0.05** → Very weak (almost no) relationship

Assignment-16

1.Analyze the regression model output and interpret the following:

a) R-squared value.

Answer-

- Definition: R-squared measures how much of the variability in the dependent variable (outcome) is explained by the independent variable(s) (predictors).
- Interpretation:
  - R-squared = 0% → The model explains none of the variability.
  - R-squared = 100% → The model explains all the variability.
- Typical Ranges:
  - Higher R-squared (e.g., above 70%) often indicates a good model fit, but context matters (in some fields like social sciences, 30-50% can still be acceptable).

b) Coefficients of the regression equation.

Answer-

- Definition: Coefficients represent the size and direction of the effect each independent variable has on the dependent variable.
- Interpretation:
  - A positive coefficient means that as the predictor increases, the dependent variable also increases (holding other variables constant).
  - A negative coefficient means that as the predictor increases, the dependent variable decreases.
- Equation form:
  - Example:

y=β0+β1x1+β2x2+…+ϵy = \beta_0 + \beta_1x_1 + \beta_2x_2 + \ldots + \epsilony=β0+β1x1+β2x2+…+ϵ

  - where:
    - β0\beta_0β0 = intercept (value of yyy when all xxx's are 0)
    - β1,β2,…\beta_1, \beta_2, \ldotsβ1,β2,… = coefficients for each predictor

c) p-values of the model terms.

- Definition: p-values test the null hypothesis that the corresponding coefficient is zero (no effect).
- Interpretation:
  - p-value $< 0.05 \rightarrow$ statistically significant (the variable meaningfully contributes to the model).
  - p-value $\geq 0.05 \rightarrow$ not statistically significant (the variable may not have a meaningful effect).
- Important: A statistically significant term implies evidence that the predictor influences the outcome.

In short:

- High R-squared = better model fit.
- Coefficients tell you the direction and strength of influence.
- Low p-values mean the variables are likely important.

Assignment-17

1.Train a regression model using a training dataset. Use it to predict outcomes for a test dataset.

Code-

```
train_data <- data.frame(
  x = c(1, 2, 3, 4, 5),
  y = c(2, 4, 5, 4, 5)
)
test_data <- data.frame(
  x = c(6, 7, 8)
)
model <- lm(y ~ x, data = train_data)
summary(model)
predictions <- predict(model, newdata = test_data)
print(predictions)
```

Output-

```
Call:
lm(formula = y ~ x, data = train_data)

Residuals:
   1    2    3    4    5
-0.8  0.6  1.0 -0.6 -0.2

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.2000     0.9381   2.345    0.101
x             0.6000     0.2828   2.121    0.124

Residual standard error: 0.8944 on 3 degrees of freedom
```

```
Multiple R-squared:    0.6,         Adjusted R-squared:  0.4667
F-statistic:   4.5 on 1 and 3 DF,  p-value: 0.124


  1   2   3
5.8 6.4 7.0
```

2.Evaluate the prediction accuracy of the model using the following metrics:

a) Mean Absolute Error (MAE).

Code-

```
actual <- c(3, 5, 2, 7)
predicted <- c(2.5, 5.5, 2.2, 6.8)
mae_manual <- mean(abs(actual - predicted))
cat("MAE (Manual Calculation):", mae_manual, "\n")
MAE <- function(actual, predicted) {
  mean(abs(actual - predicted))
}
mae_function <- MAE(actual, predicted)
cat("MAE (Using Custom Function):", mae_function, "\n")
if (!require(Metrics)) {
  install.packages("Metrics")
  library(Metrics)
} else {
  library(Metrics)
}
mae_metrics <- mae(actual, predicted)
cat("MAE (Using Metrics package):", mae_metrics, "\n")
```

Output-

```
MAE (Manual Calculation): 0.35
MAE (Using Custom Function): 0.35
MAE (Using Metrics package): 0.35
```

b) Root Mean Squared Error (RMSE).

Code-

```
actual <- c(3, 5, 2, 7)

predicted <- c(2.5, 5.5, 2.2, 6.8)

rmse_manual <- sqrt(mean((actual - predicted)^2))

cat("RMSE (Manual Calculation):", rmse_manual, "\n")

RMSE <- function(actual, predicted) {

  sqrt(mean((actual - predicted)^2))

}

rmse_function <- RMSE(actual, predicted)

cat("RMSE (Using Custom Function):", rmse_function, "\n")

rmse_metrics <- rmse(actual, predicted)

cat("RMSE (Using Metrics package):", rmse_metrics, "\n")
```

Output-

```
RMSE (Manual Calculation): 0.4123106

RMSE (Using Custom Function): 0.4123106

RMSE (Using Metrics package): 0.4123106
```

Assignment-18

1. Implement a k-Nearest Neighbors (kNN) classification model on a dataset. Evaluate the classification performance.

Code-

```
install.packages("class")     # kNN model

install.packages("caret")     # Evaluation tools

install.packages("e1071")     # Needed for confusionMatrix in caret

library(class)

library(caret)

library(e1071)

data(iris)

set.seed(123)  # For reproducibility

iris <- iris[sample(nrow(iris)), ]

normalize <- function(x) {

  return((x - min(x)) / (max(x) - min(x)))

}

iris_norm <- as.data.frame(lapply(iris[1:4], normalize))

iris_norm$Species <- iris$Species

set.seed(123)

train_index <- sample(1:nrow(iris_norm), 0.7 * nrow(iris_norm))

train_data <- iris_norm[train_index, ]

test_data <- iris_norm[-train_index, ]

train_X <- train_data[, 1:4]

test_X <- test_data[, 1:4]

train_y <- train_data$Species

test_y <- test_data$Species

k <- 3

predictions <- knn(train = train_X, test = test_X, cl = train_y, k = k)

conf_matrix <- confusionMatrix(predictions, test_y)

print(conf_matrix)
```

Output-

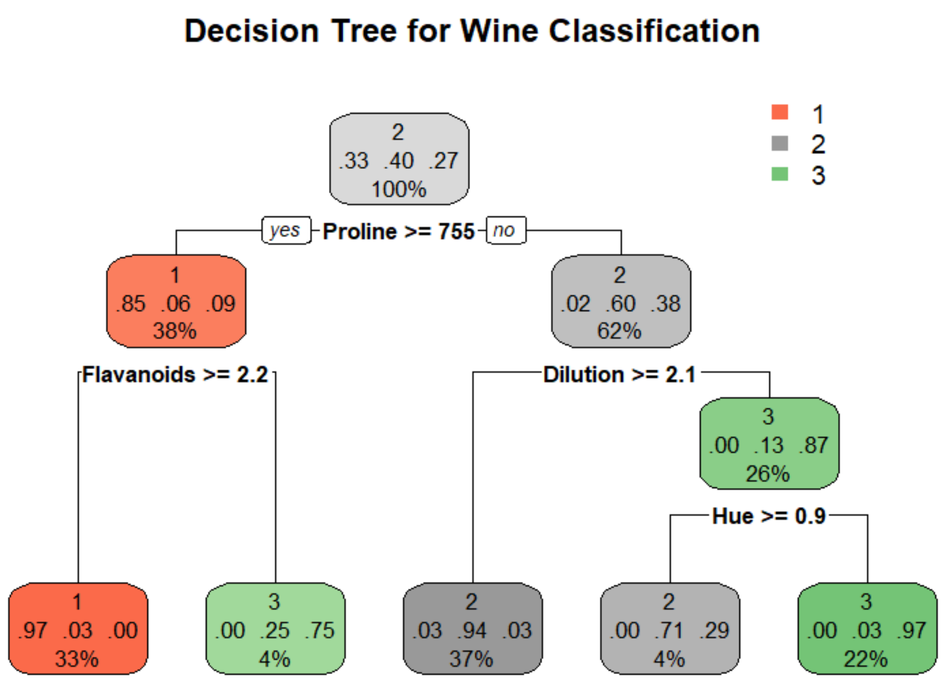
```
Confusion Matrix and Statistics


          Reference
Prediction   setosa versicolor virginica
  setosa         16          0         0
  versicolor      0          8         2
  virginica       0          1        18

Overall Statistics


               Accuracy : 0.9333
                 95% CI : (0.8173, 0.986)
    No Information Rate : 0.4444
    P-Value [Acc > NIR] : 4.158e-12


                  Kappa : 0.8961


 Mcnemar's Test P-Value : NA

Statistics by Class:


                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            0.8889           0.9000
Specificity                 1.0000            0.9444           0.9600
Pos Pred Value              1.0000            0.8000           0.9474
Neg Pred Value              1.0000            0.9714           0.9231
Prevalence                  0.3556            0.2000           0.4444
Detection Rate              0.3556            0.1778           0.4000
Detection Prevalence        0.3556            0.2222           0.4222
Balanced Accuracy           1.0000            0.9167           0.9300
```

2. Build a decision tree model using a dataset. Visualize the tree structure.

Code-

```
install.packages("rattle")

install.packages("rpart")

install.packages("rpart.plot")

library(rattle)

library(rpart)

library(rpart.plot)

data(wine, package = "rattle")

str(wine)

wine$Type <- as.factor(wine$Type)

model <- rpart(Type ~ ., data = wine, method = "class")

rpart.plot(model,

        type = 2,

        extra = 104,

        fallen.leaves = TRUE,

        main = "Decision Tree for Wine Classification")
```

Output-



3. Evaluate the classification models using the following metrics:

a) Precision.

b) Recall.

c) F1-Score.

Code-

```
install.packages("caret")
install.packages("e1071")
library(caret)
library(e1071)
set.seed(123)
index <- createDataPartition(iris$Species, p = 0.7, list = FALSE)
train_data <- iris[index, ]
test_data <- iris[-index, ]
library(rpart)
model <- rpart(Species ~ ., data = train_data, method = "class")
predictions <- predict(model, newdata = test_data, type = "class")
conf_mat <- confusionMatrix(predictions, test_data$Species)
print(conf_mat)
cm <- table(Predicted = predictions, Actual = test_data$Species)
precision <- diag(cm) / colSums(cm)
recall <- diag(cm) / rowSums(cm)
f1 <- 2 * (precision * recall) / (precision + recall)
metrics <- data.frame(Precision = precision,
                      Recall = recall,
                      F1_Score = f1)
print(metrics)
```

Output-

```
Confusion Matrix and Statistics


          Reference
```

```
Prediction   setosa versicolor virginica
  setosa         15          0         0
  versicolor      0         14         2
  virginica       0          1        13

Overall Statistics

               Accuracy : 0.9333
                 95% CI : (0.8173, 0.986)
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            0.9333           0.8667
Specificity                 1.0000            0.9333           0.9667
Pos Pred Value              1.0000            0.8750           0.9286
Neg Pred Value              1.0000            0.9655           0.9355
Prevalence                  0.3333            0.3333           0.3333
Detection Rate              0.3333            0.3111           0.2889
Detection Prevalence        0.3333            0.3556           0.3111
Balanced Accuracy           1.0000            0.9333           0.9167

           Precision    Recall  F1_Score
  setosa     1.0000000 1.0000000 1.0000000
  versicolor 0.9333333 0.8750000 0.9032258
  virginica  0.8666667 0.9285714 0.8965517
```

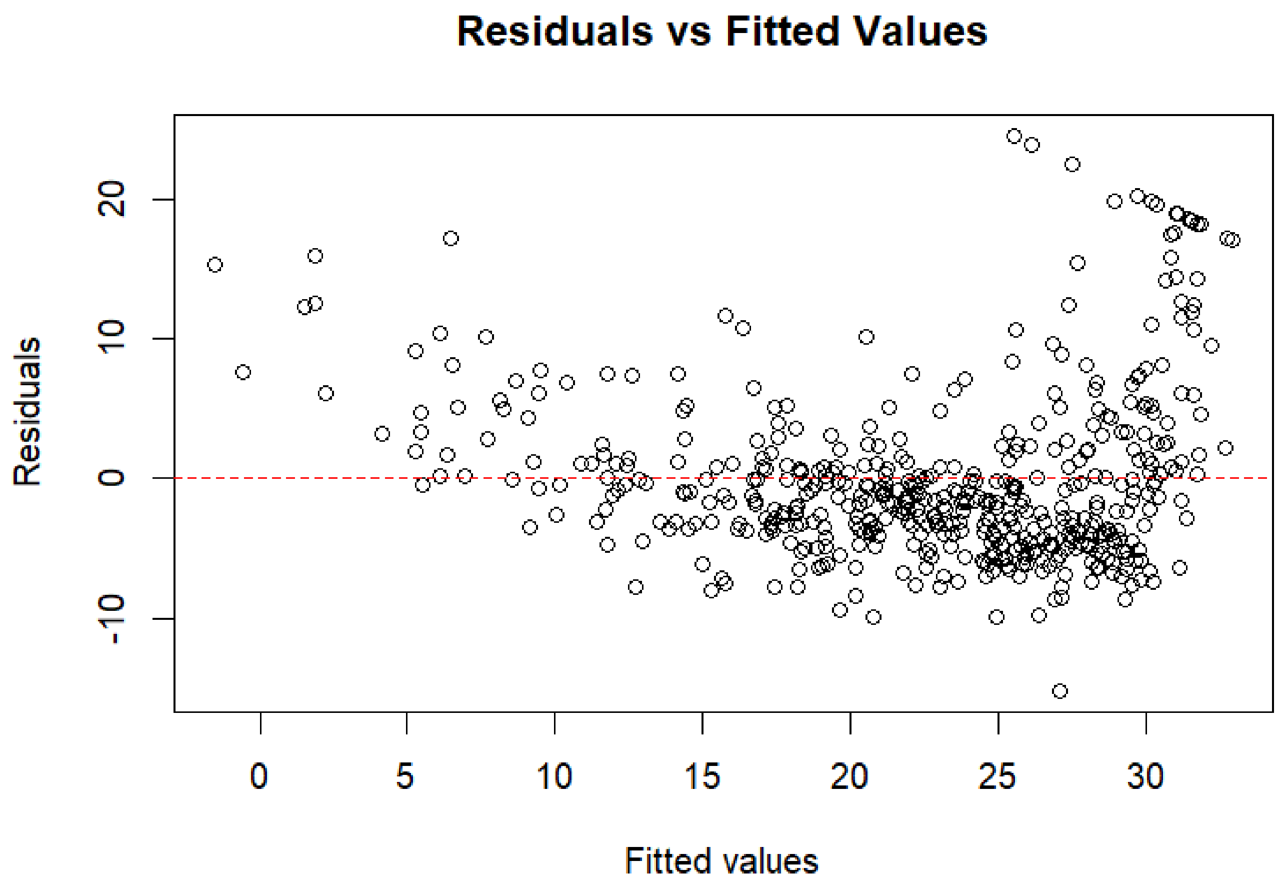
Assignment-19

1.Use the Boston dataset from the MASS package to predict medv (median home value) based on lstat (percentage of lower status population). Evaluate the residuals and R-squared value.

Code-

```
install.packages("MASS
library(MASS)
data("Boston")
head(Boston)
model <- lm(medv ~ lstat, data = Boston)
summary(model)
r_squared <- summary(model)$r.squared
cat("R-squared:", r_squared, "\n")
plot(Boston$lstat, Boston$medv,
    xlab = "LSTAT (% lower status population)",
    ylab = "MEDV (Median home value)",
    main = "Linear Regression: MEDV ~ LSTAT",
    pch = 20, col = "blue")
abline(model, col = "red", lwd = 2)

# Step 7: Evaluate residuals
residuals <- resid(model)
head(residuals)

# Step 8: Plot residuals vs fitted values
plot(model$fitted.values, residuals,
    xlab = "Fitted values",
    ylab = "Residuals",
    main = "Residuals vs Fitted Values")
abline(h = 0, col = "red", lty = 2)
```

Output-


Residuals vs Fitted Values
Residuals
Fitted values
20
10
0
-10
0
5
10
15
20
25
30

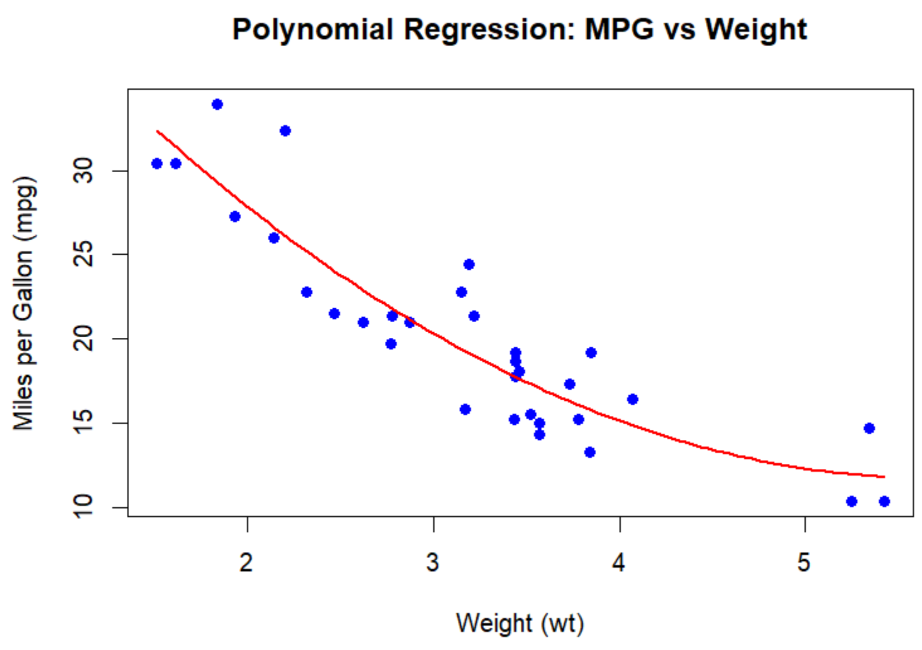
Assignment-20

Fit a polynomial regression model to predict mpg based on wt and visualize the results with a smooth curve.

Code-

```
data(mtcars)  # Load the mtcars dataset (contains mpg and wt)
model_poly <- lm(mpg ~ poly(wt, 2), data = mtcars)
summary(model_poly)
wt_seq <- seq(min(mtcars$wt), max(mtcars$wt), length.out = 100)
mpg_pred <- predict(model_poly, newdata = data.frame(wt = wt_seq))
plot(mtcars$wt, mtcars$mpg,
    xlab = "Weight (wt)",
    ylab = "Miles per Gallon (mpg)",
    main = "Polynomial Regression: MPG vs Weight",
    pch = 16, col = "blue")
lines(wt_seq, mpg_pred, col = "red", lwd = 2)
```

Output-



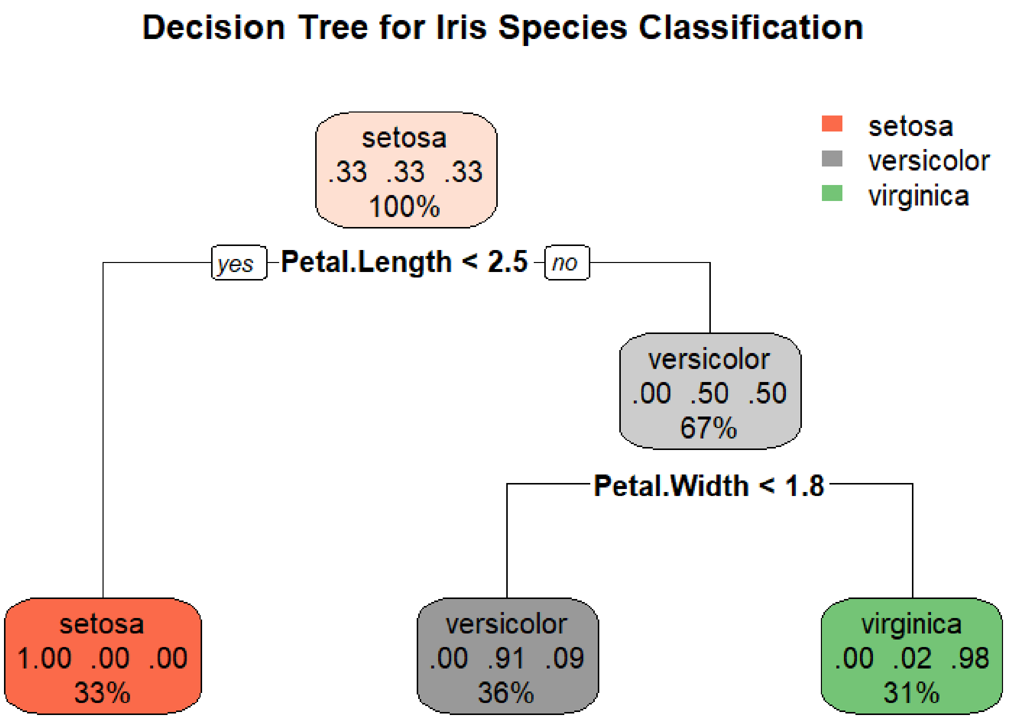
Assignment-21

1.Use the rpart package to build a decision tree classifier for the iris dataset, predicting the species of flowers.

Code-

```
install.packages("rpart")

install.packages("rpart.plot")

library(rpart)

library(rpart.plot)

data(iris)

model <- rpart(Species ~ ., data = iris, method = "class")

summary(model)

rpart.plot(model,

        type = 2,           # Labels all nodes

        extra = 104,         # Displays class probabilities and percentages

        fallen.leaves = TRUE,

        main = "Decision Tree for Iris Species Classification")
```

Output-


Decision Tree for Iris Species Classification
setosa
versicolor
virginica
setosa .33 .33 .33 100%
yes
Petal.Length < 2.5
no
versicolor .00 .50 .50 67%
Petal.Width < 1.8
setosa 1.00 .00 .00 33%
versicolor .00 .91 .09 36%
virginica .00 .02 .98 31%

Assignment-22

1.Use the class package to build a KNN classifier for the iris dataset. Experiment with different values of k.

Code-

```
install.packages("class")
install.packages("caret")   # for confusion matrix and performance metrics
library(class)
library(caret)
data(iris)
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}
iris_norm <- as.data.frame(lapply(iris[, 1:4], normalize))
iris_norm$Species <- iris$Species  # Append target variable back
set.seed(123)
train_index <- createDataPartition(iris_norm$Species, p = 0.7, list = FALSE)
train_data <- iris_norm[train_index, ]
test_data <- iris_norm[-train_index, ]
train_x <- train_data[, 1:4]
train_y <- train_data$Species
test_x <- test_data[, 1:4]
test_y <- test_data$Species
for (k in c(1, 3, 5, 7, 9)) {
  cat("\nResults for k =", k, "\n")
  pred <- knn(train = train_x, test = test_x, cl = train_y, k = k)
  cm <- confusionMatrix(pred, test_y)
  print(cm$table)
  cat("Accuracy:", round(cm$overall['Accuracy'], 4), "\n")
}
```

Output-

```
Results for k = 1
          Reference
Prediction   setosa versicolor virginica
  setosa        15          0         0
  versicolor     0         14         2
  virginica      0          1        13
Accuracy: 0.9333

Results for k = 3
          Reference
Prediction   setosa versicolor virginica
  setosa        15          0         0
  versicolor     0         14         2
  virginica      0          1        13
Accuracy: 0.9333

Results for k = 5
          Reference
Prediction   setosa versicolor virginica
  setosa        15          0         0
  versicolor     0         14         2
  virginica      0          1        13
Accuracy: 0.9333

Results for k = 7
          Reference
Prediction   setosa versicolor virginica
  setosa        15          0         0
  versicolor     0         15         2
  virginica      0          0        13
Accuracy: 0.9556
```

```
Results for k = 9
          Reference
Prediction   setosa versicolor virginica
  setosa        15          0         0
  versicolor     0         15         1
  virginica      0          0        14
Accuracy: 0.9778
```

Assignment-23

Compare classification models (e.g., logistic regression, decision tree, SVM) on the iris dataset using metrics such as accuracy, precision, recall, and F1 score.

Code-

```
install.packages("nnet")
install.packages("rpart")
install.packages("e1071")
install.packages("caret")
library(nnet)
library(rpart)
library(e1071)
library(caret)
data(iris)
set.seed(123)
train_index <- createDataPartition(iris$Species, p = 0.7, list = FALSE)
train_data <- iris[train_index, ]
test_data <- iris[-train_index, ]
log_model <- multinom(Species ~ ., data = train_data)
tree_model <- rpart(Species ~ ., data = train_data, method = "class")
svm_model <- svm(Species ~ ., data = train_data)
log_pred <- predict(log_model, test_data)
tree_pred <- predict(tree_model, test_data, type = "class")
svm_pred <- predict(svm_model, test_data)
evaluate_model <- function(pred, actual) {
  cm <- confusionMatrix(pred, actual)
  list(
    Accuracy = cm$overall['Accuracy'],
    Precision = cm$byClass[, 'Precision'],
    Recall = cm$byClass[, 'Recall'],
    F1 = cm$byClass[, 'F1']
  )
}
log_metrics <- evaluate_model(log_pred, test_data$Species)
tree_metrics <- evaluate_model(tree_pred, test_data$Species)
svm_metrics <- evaluate_model(svm_pred, test_data$Species)
cat("◈ Logistic Regression Metrics:\n")
print(log_metrics)
cat("\n◈ Decision Tree Metrics:\n")
print(tree_metrics)
cat("\n◈ SVM Metrics:\n")
print(svm_metrics)
```

Output-

```
# weights:  18 (10 variable)
initial  value 115.354290
iter  10 value 11.160147
iter  20 value 3.325479
iter  30 value 2.713763
iter  40 value 2.623668
iter  50 value 2.409177
iter  60 value 2.346975
iter  70 value 2.246318
iter  80 value 2.223840
iter  90 value 1.930046
iter 100 value 1.883252
final  value 1.883252
stopped after 100 iterations

◈ Logistic Regression Metrics:
$Accuracy
 Accuracy
```

```
0.9555556

$Precision
   Class: setosa Class: versicolor  Class: virginica
      1.0000000        0.8823529        1.0000000

$Recall
   Class: setosa Class: versicolor  Class: virginica
      1.0000000        1.0000000        0.8666667

$F1
   Class: setosa Class: versicolor  Class: virginica
      1.0000000        0.9375000        0.9285714
```

◈ Decision Tree Metrics:

```
$Accuracy
 Accuracy
0.9333333

$Precision
   Class: setosa Class: versicolor  Class: virginica
      1.0000000        0.8750000        0.9285714

$Recall
   Class: setosa Class: versicolor  Class: virginica
      1.0000000        0.9333333        0.8666667

$F1
   Class: setosa Class: versicolor  Class: virginica
      1.0000000        0.9032258        0.8965517
```

◈ SVM Metrics:

```
$Accuracy
 Accuracy
0.9333333

$Precision
    Class: setosa Class: versicolor  Class: virginica
        1.0000000         0.8750000         0.9285714

$Recall
    Class: setosa Class: versicolor  Class: virginica
        1.0000000         0.9333333         0.8666667

$F1
    Class: setosa Class: versicolor  Class: virginica
        1.0000000         0.9032258         0.8965517
```