LAB EXERCISE 2

R PROGRAMMING

NAME : KRISHVANTH KUMAR E

REG NO:192125063

Set-I

1. .(i) Write a function called kelvin_to_celsius() that takes a temperature in Kelvin and returns that temperature in Celsius (Hint: To convert from Kelvin to Celsius you subtract 273.15)

(ii) Write suitable R code to compute the mean, median ,mode of the following values

c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)

(iii) Write R code to find 2nd

highest and 3rd Lowest value of above problem.

CODE

```
kelvin_to_celsius <- function(temperature_in_kelvin) {
  return(temperature_in_kelvin - 273.15)
}
kelvin_temperature <- 300
celsius_temperature <- kelvin_to_celsius(kelvin_temperature)
print(paste(kelvin_temperature, "Kelvin is equal to", celsius_temperature, "Celsius."))
values <- c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)
mean_value <- mean(values)
print(paste("Mean:", mean_value))
median_value <- median(values)
print(paste("Median:", median_value))
mode_value <- names(table(values))[table(values) == max(table(values))]
print(paste("Mode:", mode_value))
```

LAB EXERCISE 2

R PROGRAMMING

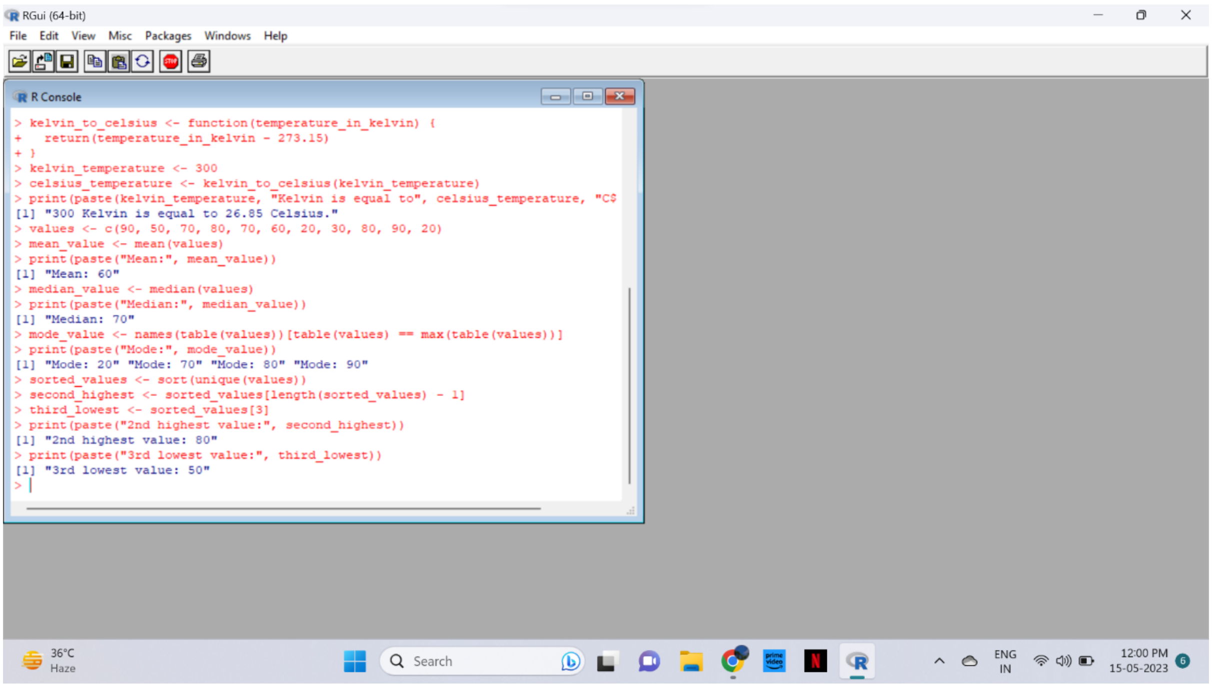
sorted_values <- sort(unique(values))

second_highest <- sorted_values[length(sorted_values) - 1]

third_lowest <- sorted_values[3]

print(paste("2nd highest value:", second_highest))

print(paste("3rd lowest value:", third_lowest))



2. Explore the airquality dataset. It contains daily air quality measurements from New York during a period

of five months:

- Ozone: mean ozone concentration (ppb),
- Solar.R: solar radiation (Langley),
- Wind: average wind speed (mph),
- Temp: maximum daily temperature in degrees Fahrenheit,
- Month: numeric month (May=5, June=6, and so on),
- Day: numeric day of the month (1-31).

i. Compute the mean temperature(don’t use build in function)

LAB EXERCISE 2

R PROGRAMMING

ii. Extract the first five rows from airquality.

iii. Extract all columns from airquality except Temp and Wind

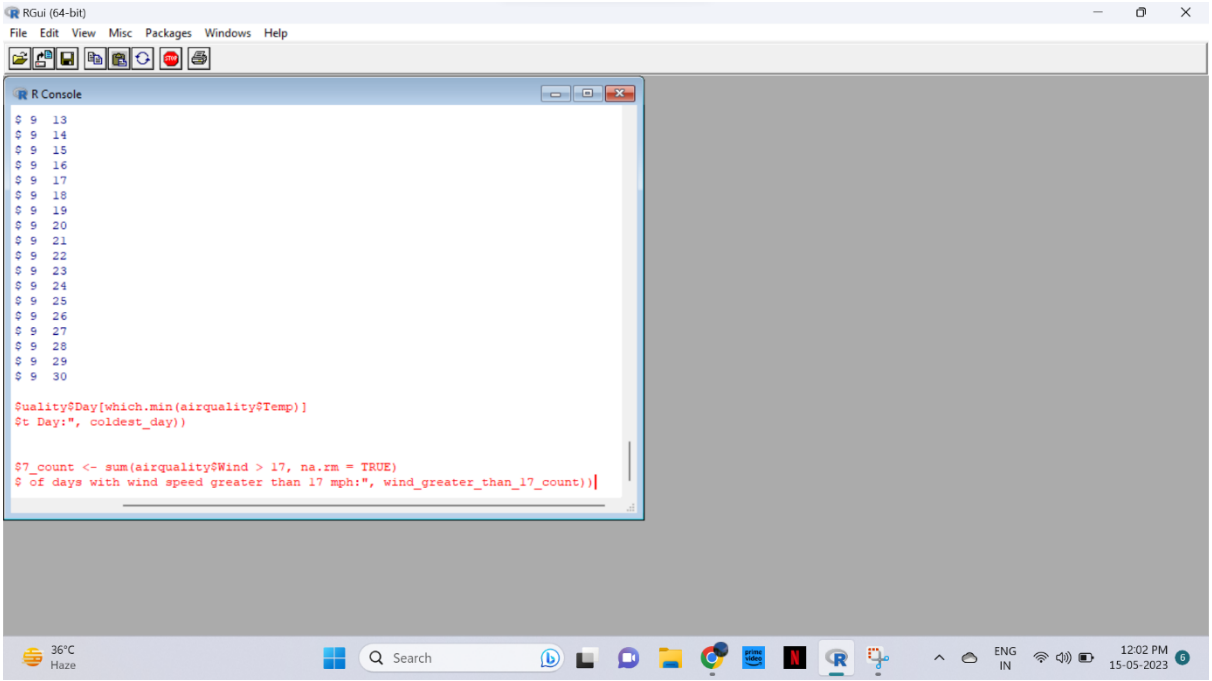
iv. Which was the coldest day during the period?

v. How many days was the wind speed greater than 17 mph?

```
data(airquality)

mean_temperature <- sum(airquality$Temp) / length(airquality$Temp)

print(paste("Mean Temperature:", mean_temperature))


first_five_rows <- airquality[1:5, ]

print("First five rows:")

print(first_five_rows)


selected_columns <- airquality[, !(names(airquality) %in% c("Temp", "Wind"))]

print("Selected columns:")

print(selected_columns)


coldest_day <- airquality$Day[which.min(airquality$Temp)]

print(paste("Coldest Day:", coldest_day))


wind_greater_than_17_count <- sum(airquality$Wind > 17, na.rm = TRUE)

print(paste("Number of days with wind speed greater than 17 mph:",
wind_greater_than_17_count))
```

LAB EXERCISE 2

R PROGRAMMING



3. (i) Get the Summary Statistics of air quality dataset

(ii)Melt airquality data set and display as a long – format data?

(iii)Melt airquality data and specify month and day to be "ID variables"?

(iv)Cast the molten airquality data set with respect to month and date features

(v) Use cast function appropriately and compute the average of Ozone, Solar.R , Wind and temperature per month?

CODE

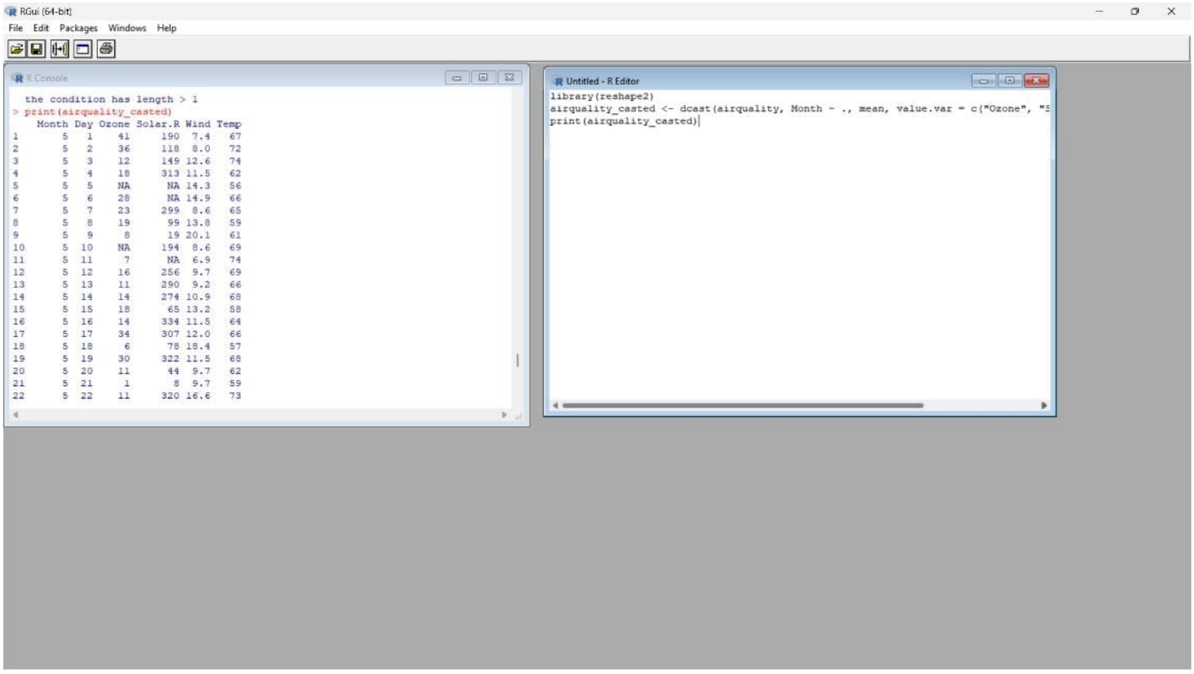
```
summary(airquality)
library(reshape2)
melted_data <- melt(airquality)
print(melted_data)
melted_data <- melt(airquality, id.vars = c("Month", "Day"))
print(melted_data)
cast_data <- dcast(melted_data, Month + Day ~ variable)
print(cast_data)
```

LAB EXERCISE 2

R PROGRAMMING

cast_data <- dcast(melted_data, Month ~ variable, mean)

print(cast_data)



4.(i) Find any missing values(na) in features and drop the missing values if its less than 10% else replace that with mean of that feature.

(ii) Apply a linear regression algorithm using Least Squares Method on “Ozone” and “Solar.R”

(iii)Plot Scatter plot between Ozone and Solar and add regression line created by above model

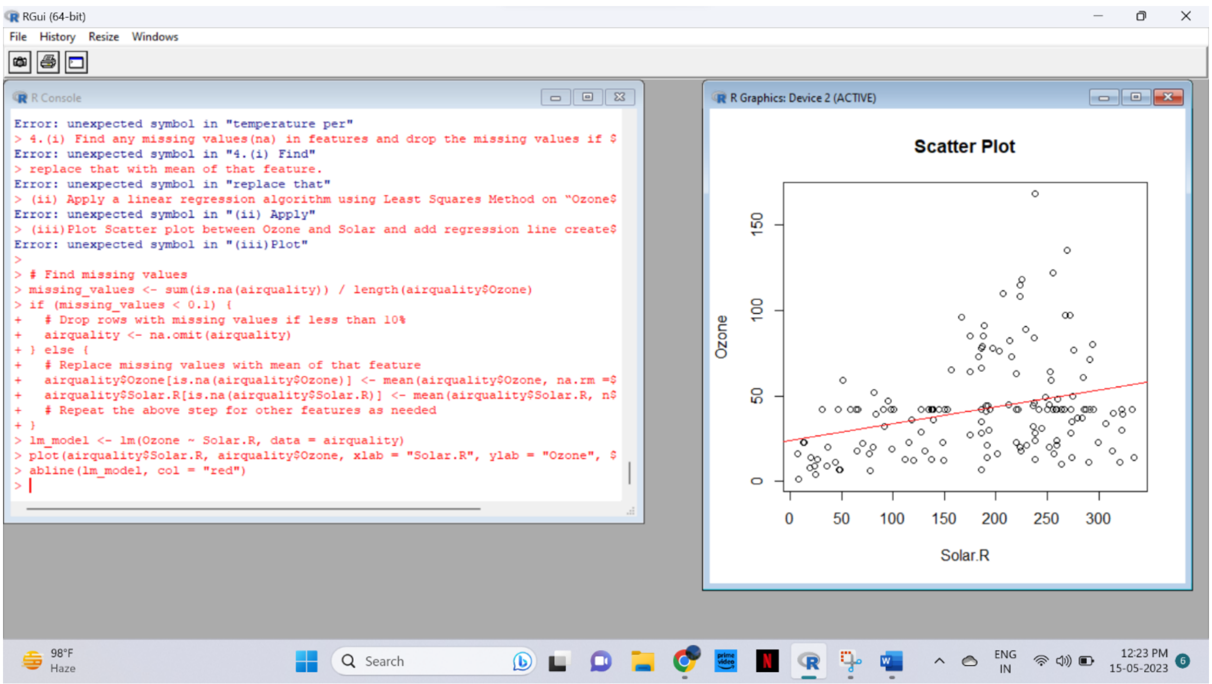
CODE

```
 # Find missing values
missing_values <- sum(is.na(airquality)) / length(airquality$Ozone)
if (missing_values < 0.1) {
  # Drop rows with missing values if less than 10%
```

LAB EXERCISE 2

R PROGRAMMING

```
 airquality <- na.omit(airquality)
} else {
 # Replace missing values with mean of that feature
 airquality$Ozone[is.na(airquality$Ozone)] <- mean(airquality$Ozone, na.rm = TRUE)
 airquality$Solar.R[is.na(airquality$Solar.R)] <- mean(airquality$Solar.R, na.rm = TRUE)
 # Repeat the above step for other features as needed
}
lm_model <- lm(Ozone ~ Solar.R, data = airquality)
plot(airquality$Solar.R, airquality$Ozone, xlab = "Solar.R", ylab = "Ozone", main = "Scatter Plot")
abline(lm_model, col = "red")
```



LAB EXERCISE 2

R PROGRAMMING

1. (i)Write a function to find the factorial of a given number using "for" Loop

(ii) Create a 3x4 matrix with 12 random numbers between 1-100; have the matrix be filled our row-by-row, instead of column-by-column. Name the columns of the matrix uno, dos, tres, cuatro, and

the rows x, y, z. Scale the matrix by 10 and save the result.

(iii) Extract the column called "uno" as a vector from the original matrix and save the result

CODE

```
factorial <- function(n) {
  result <- 1
  for (i in 1:n) {
    result <- result * i
  }
  return(result)
}
number <- 5
factorial_of_number <- factorial(number)
print(factorial_of_number)
set.seed(123)  # Setting a seed for reproducibility

# Create the matrix
matrix_data <- matrix(sample(1:100, 12), nrow = 3, ncol = 4, byrow = TRUE)

# Scale the matrix by 10
scaled_matrix <- matrix_data * 10

# Name the columns and rows
```

LAB EXERCISE 2

R PROGRAMMING

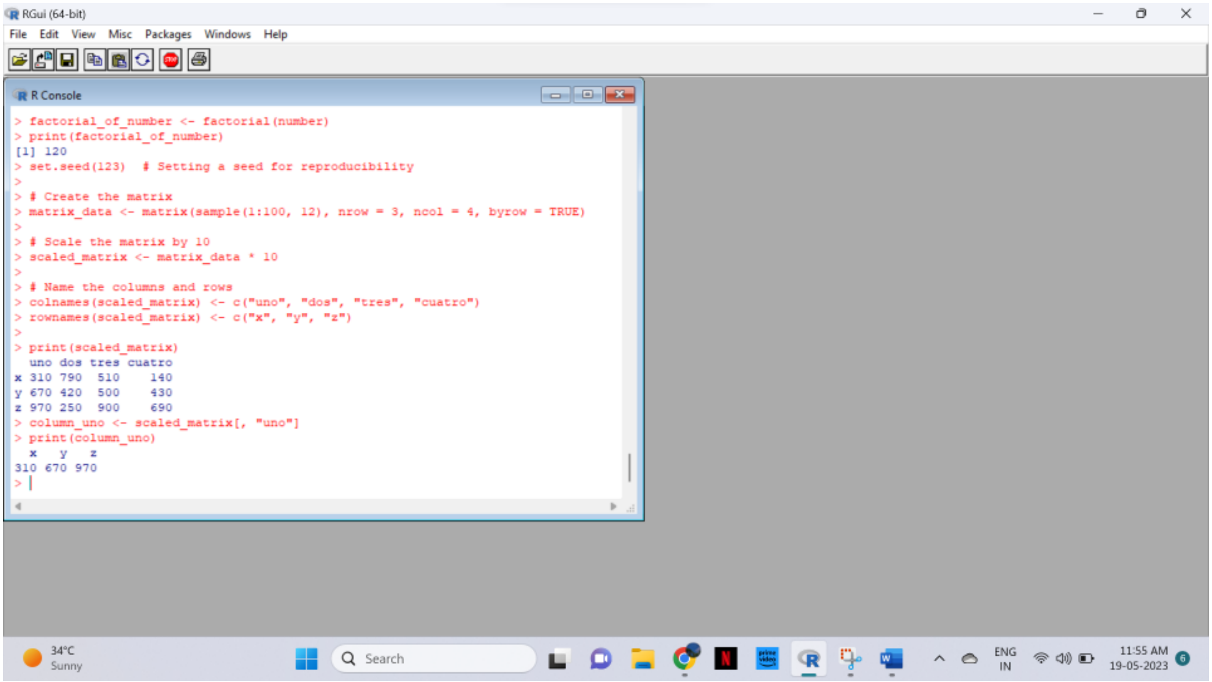
colnames(scaled_matrix) <- c("uno", "dos", "tres", "cuatro")

rownames(scaled_matrix) <- c("x", "y", "z")

print(scaled_matrix)

column_uno <- scaled_matrix[, "uno"]

print(column_uno)



2. In 1936, Edgar Anderson collected data to quantify the geographic variations of iris flowers. The data

set consists of 50 samples from each of the three sub-species (iris setosa, iris virginica, and iris

versicolor).Four features were measured in centimeters (cm): the lengths and the widths of both

sepals and petals

(i)Find dimension, Structure, Summary statistics, Standard Deviation of all features.

(ii)Find mean and standard deviation of features groped by three species of Iris

flowers (Iris setosa, Iris virginica and Iris versicolor)

(iii)Find quantile value of sepal width and length

LAB EXERCISE 2

R PROGRAMMING

(iv)create new data frame named iris1 which have a new column name

Sepal.Length.Cate that categorizes “Sepal.Length” by quantile

(v) Average value of numerical varialbes by two categorical variables: Species and

Sepal.Length.Cate.

CODE

```
# Load the iris dataset
data(iris)

# Dimension of the dataset
dimension <- dim(iris)
print(dimension)

# Structure of the dataset
structure <- str(iris)

# Summary statistics
summary_stats <- summary(iris)

# Standard deviation of all features
std_dev <- apply(iris[, 1:4], 2, sd)
print(std_dev)
# Mean and standard deviation grouped by species
mean_species <- aggregate(iris[, 1:4], by = list(iris$Species), FUN = mean)
sd_species <- aggregate(iris[, 1:4], by = list(iris$Species), FUN = sd)
print(mean_species)
print(sd_species)
# Quantile values of sepal width and length
```

LAB EXERCISE 2

R PROGRAMMING

quantile_width <- quantile(iris$Sepal.Width)

quantile_length <- quantile(iris$Sepal.Length)

print(quantile_width)

print(quantile_length)

Create new data frame

iris1 <- iris

quantile_thresholds <- quantile(iris$Sepal.Length, probs = c(0, 0.25, 0.5, 0.75, 1))

categories <- cut(iris$Sepal.Length, breaks = quantile_thresholds, labels = c("Q1", "Q2", "Q3", "Q4"))

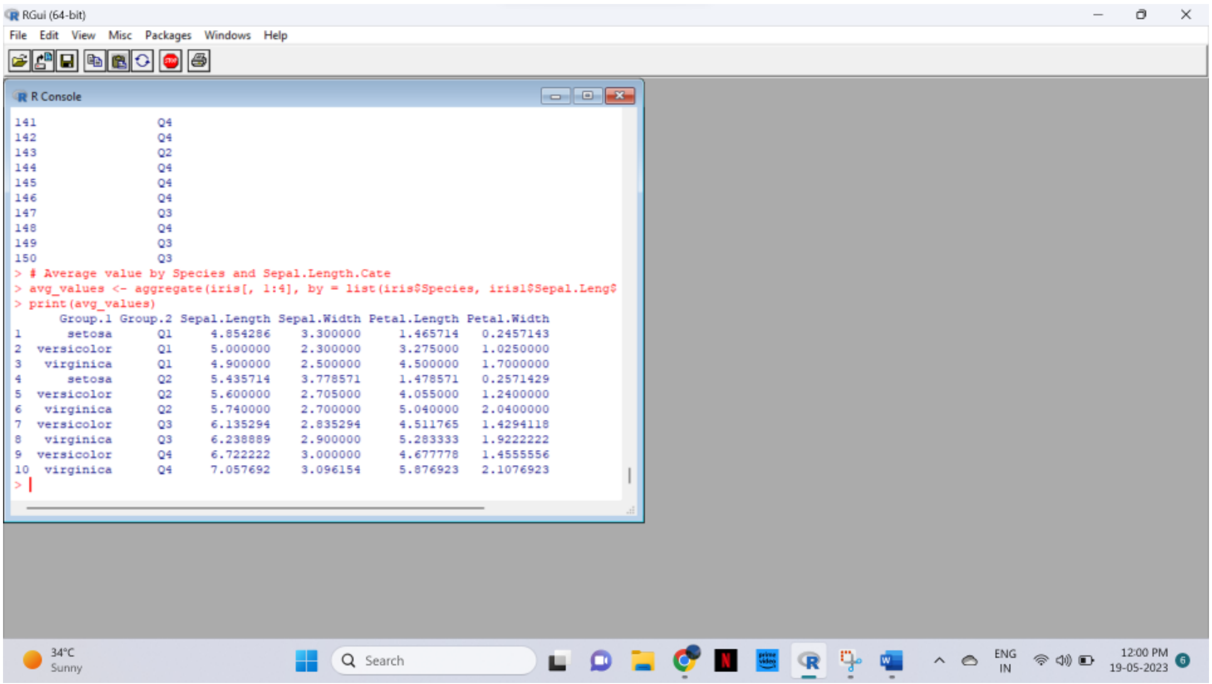
iris1$Sepal.Length.Cate <- categories

print(iris1)

Average value by Species and Sepal.Length.Cate

avg_values <- aggregate(iris[, 1:4], by = list(iris$Species, iris1$Sepal.Length.Cate), FUN = mean)

print(avg_values)



3. (i)Plot Scatter plot between sepals width and length grouped by Species

(ii) Plot Scatter plot between petals width and length grouped by Species

LAB EXERCISE 2

R PROGRAMMING

(iii)Draw the Box plot for Sepals length grouped by Species

(iv) Draw the Box plot for petals length grouped by Species

(v)Find the correlation among the four features

CODE

i.)library(ggplot2)

Scatter plot for Sepal Width vs. Length grouped by Species

ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) +

  geom_point() +

  labs(x = "Sepal Width", y = "Sepal Length", title = "Scatter plot of Sepal Width vs. Length grouped by Species")

ii.)# Scatter plot for Petal Width vs. Length grouped by Species

ggplot(iris, aes(x = Petal.Width, y = Petal.Length, color = Species)) +

  geom_point() +

  labs(x = "Petal Width", y = "Petal Length", title = "Scatter plot of Petal Width vs. Length grouped by Species")

iii).# Box plot for Sepal Length grouped by Species

ggplot(iris, aes(x = Species, y = Sepal.Length)) +

  geom_boxplot() +

  labs(x = "Species", y = "Sepal Length", title = "Box plot of Sepal Length grouped by Species")

iv.)# Box plot for Petal Length grouped by Species

ggplot(iris, aes(x = Species, y = Petal.Length)) +

  geom_boxplot() +

  labs(x = "Species", y = "Petal Length", title = "Box plot of Petal Length grouped by Species")

V.)

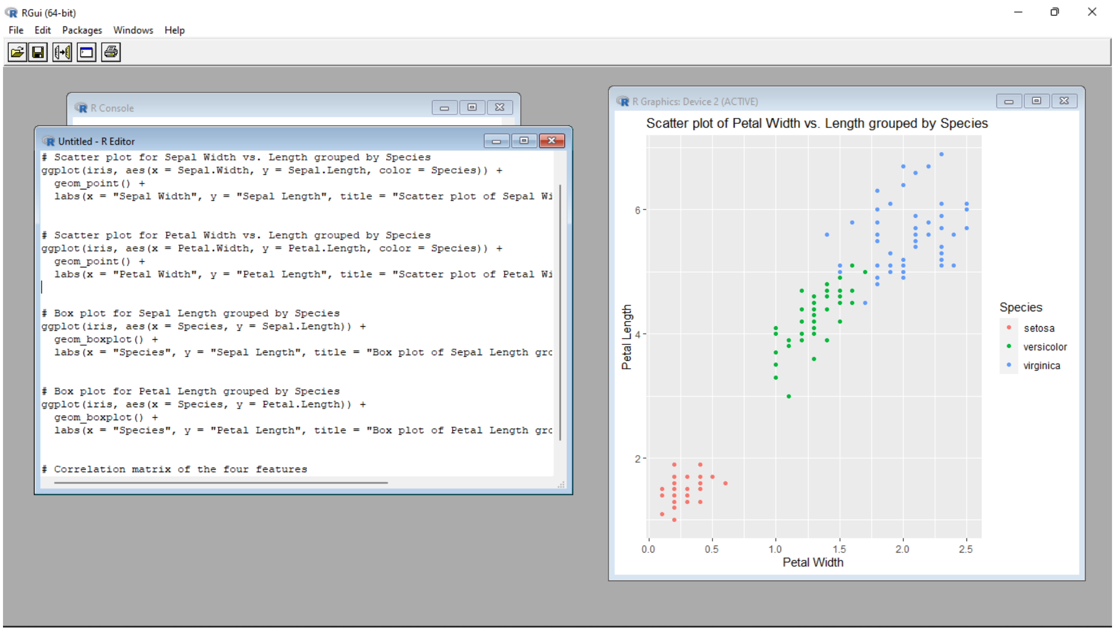
Correlation matrix of the four features

cor(iris[, c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")])

Output:

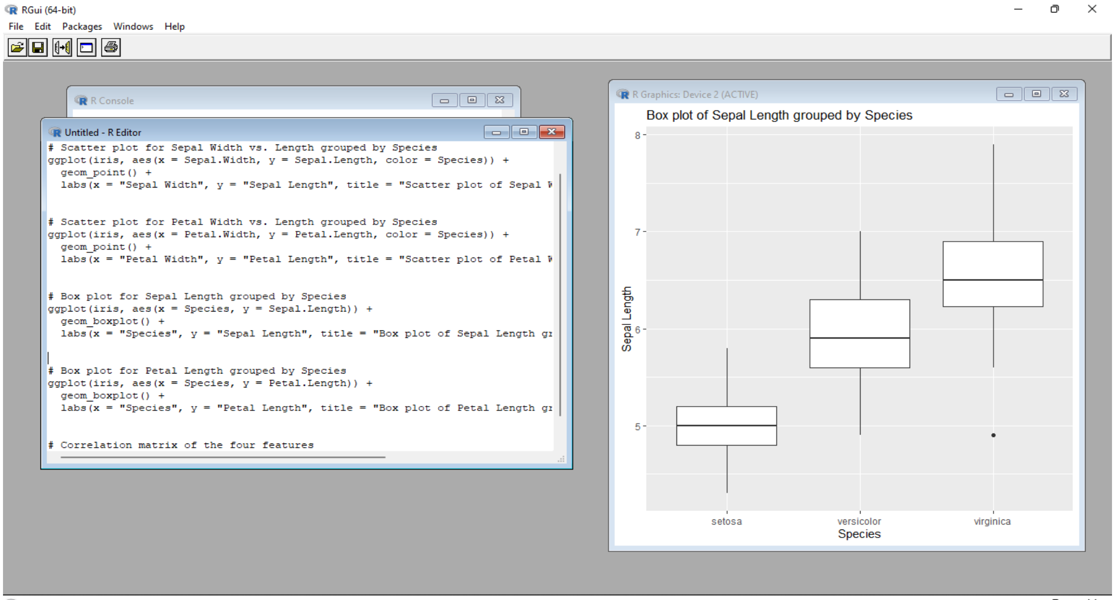
LAB EXERCISE 2

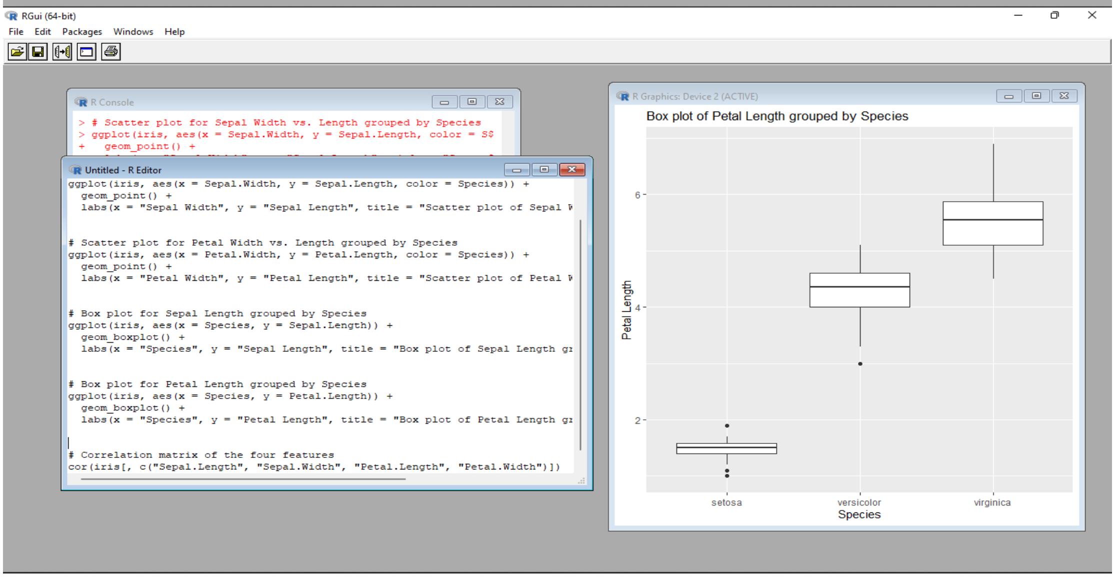
R PROGRAMMING



LAB EXERCISE 2

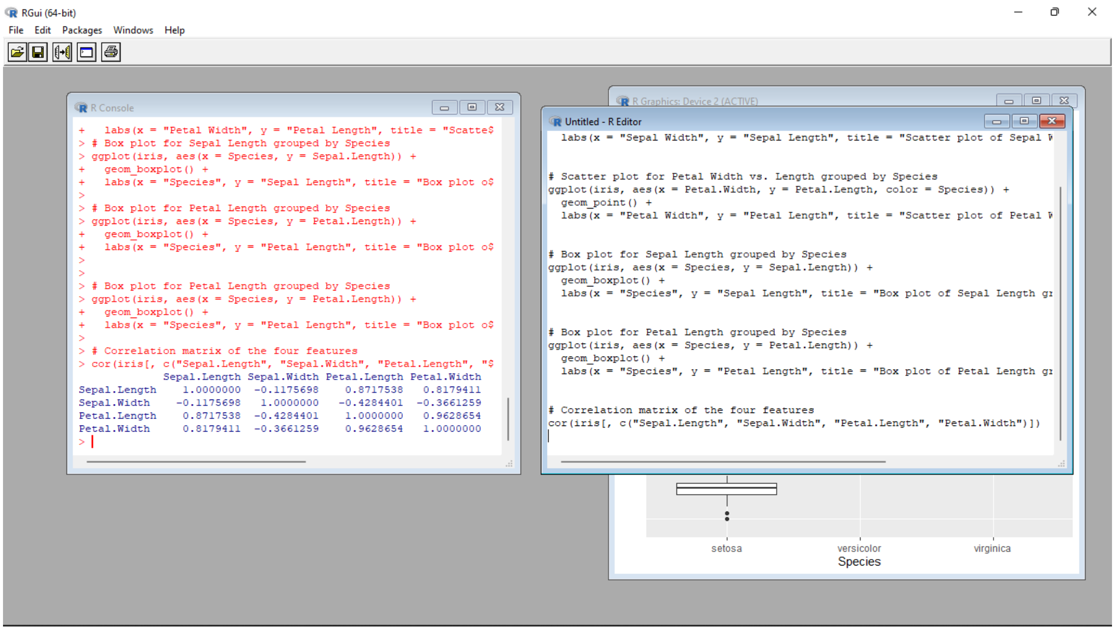
R PROGRAMMING





LAB EXERCISE 2

R PROGRAMMING



4.(i) Randomly Sample the iris dataset such as 50% data for training and 50% for test

(ii)find summary statistics of above train and test dataset.

(iii)Create Logistics regression with train data

(iv)Predict the probability of the model using test data

(v)Create Confusion matrix for above test model

CODE

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


# Randomly sample the iris dataset
train_indices <- sample(1:nrow(iris), nrow(iris) * 0.5)
train_data <- iris[train_indices, ]
test_data <- iris[-train_indices, ]
```

LAB EXERCISE 2

R PROGRAMMING

ii.)# Summary statistics of train dataset

summary(train_data)

Summary statistics of test dataset

summary(test_data)

iii.)# Create logistic regression model

model <- glm(Species ~ ., data = train_data, family = binomial)

print(model)

iv.)# Predict probabilities using the test data

probabilities <- predict(model, newdata = test_data, type = "response")

print(probabilities)

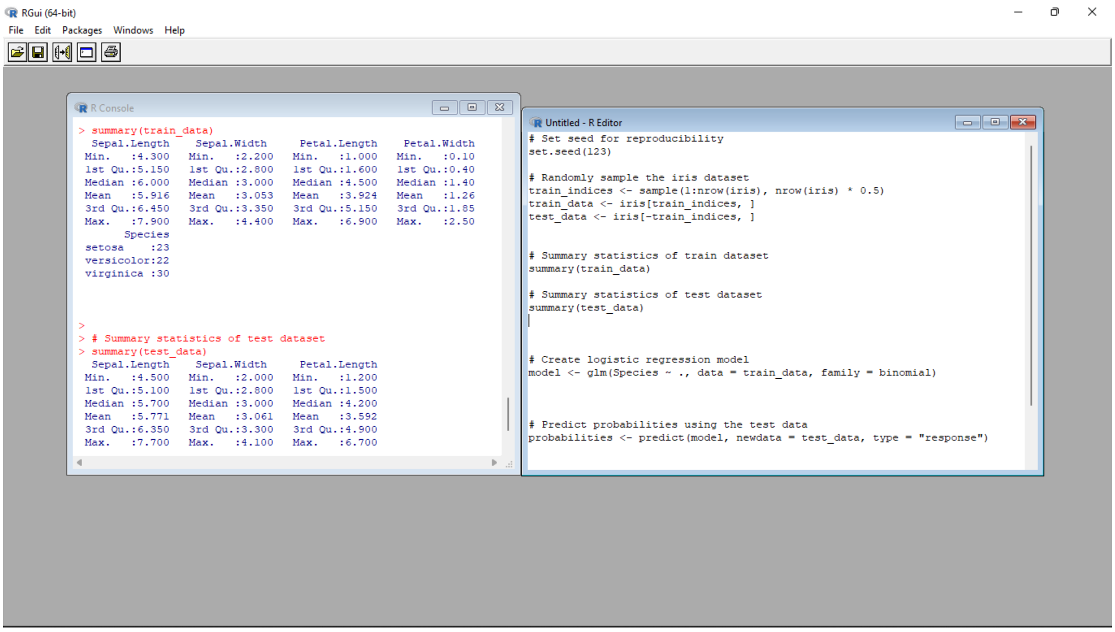
V.)# Create confusion matrix

LAB EXERCISE 2

R PROGRAMMING

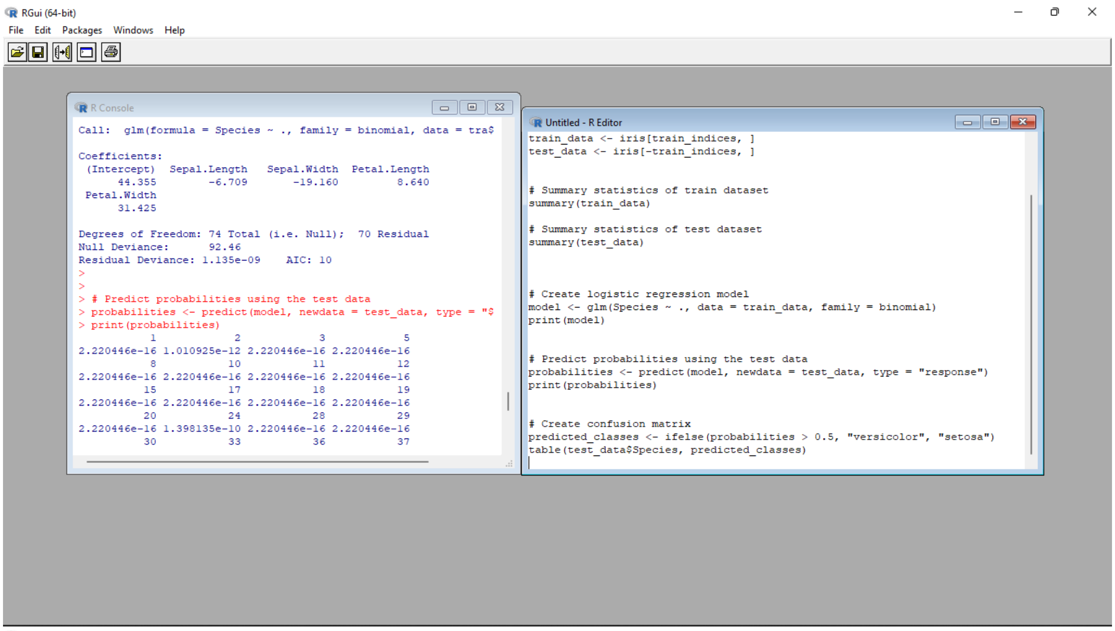
```
predicted_classes <- ifelse(probabilities > 0.5, "versicolor", "setosa")
table(test_data$Species, predicted_classes)
```

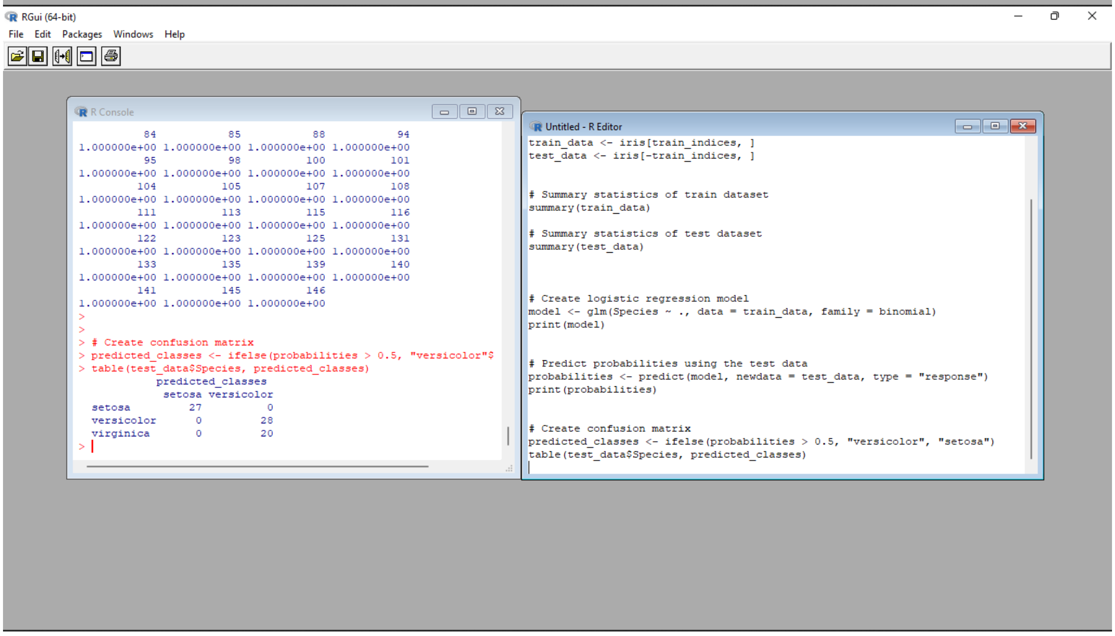
Output :



LAB EXERCISE 2

R PROGRAMMING





Set-III

1. Suppose you track your commute times for two weeks (10 days) and you find the following

times in minutes 17 16 20 24 22 15 21 15 17 22 Enter this into R as vector data type.

(i)create function maxi to find the longest commute time, the function avger to find the average and the function mini to find the minimum.

(ii)Oops, the 24 was a mistake. It should have been 18. How can you fix this? Do so, and then find

the new average using above functions.

(iii)How many times was your commute 20 minutes or more?

Input:

```
# Vector of commute times
commute_times <- c(17, 16, 20, 24, 22, 15, 21, 15, 17, 22)

# Function to find the longest commute time
maxi <- function(commute_times) {
  max(commute_times)
}

# Function to find the average commute time
avger <- function(commute_times) {
  mean(commute_times)
}

# Function to find the minimum commute time
mini <- function(commute_times) {
  min(commute_times)
}

# Calling the functions
max_time <- maxi(commute_times)
```

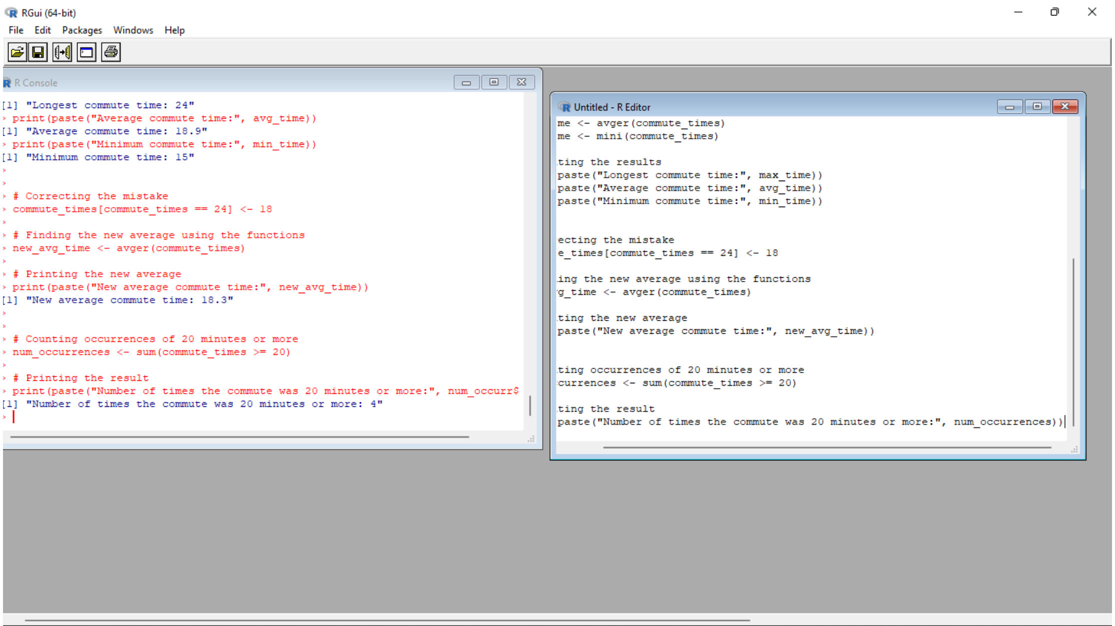
LAB EXERCISE 2

R PROGRAMMING

```
avg_time <- avger(commute_times)
min_time <- mini(commute_times)

# Printing the results
print(paste("Longest commute time:", max_time))
print(paste("Average commute time:", avg_time))
print(paste("Minimum commute time:", min_time))



# Correcting the mistake
commute_times[commute_times == 24] <- 18

# Finding the new average using the functions
new_avg_time <- avger(commute_times)

# Printing the new average
print(paste("New average commute time:", new_avg_time))



# Counting occurrences of 20 minutes or more
num_occurrences <- sum(commute_times >= 20)

# Printing the result
print(paste("Number of times the commute was 20 minutes or more:", num_occurrences))
```

LAB EXERCISE 2

R PROGRAMMING



2. There is a popular built-in data set in R called "**mtcars**" (Motor Trend Car Road Tests), which is retrieved from the 1974 Motor Trend US Magazine.

(i)Find the dimension of the data set

(ii)Give the statistical summary of the features.

(iii)Find the largest and smallest value of the variable hp (horsepower).

(iv)Give the mean of mileage per gallon (mpg) with respect to transmission model (feature named

as 'am')

(v)Give the median of horsepower (hp) with respect to cylinder displacement(cyl)

Input:

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


# Get the dimension of the dataset
dim(mtcars)
```

LAB EXERCISE 2

R PROGRAMMING

```
# Get the summary of the dataset
summary(mtcars)



# Find the largest value of "hp"
max_hp <- max(mtcars$hp)

# Find the smallest value of "hp"
min_hp <- min(mtcars$hp)

# Print the results
print(paste("Largest value of horsepower (hp):", max_hp))
print(paste("Smallest value of horsepower (hp):", min_hp))



# Calculate the mean of mpg by transmission type (am)
mean_mpg <- tapply(mtcars$mpg, mtcars$am, mean)

# Print the mean mpg for each transmission type
print("Mean mpg by transmission type:")
print(mean_mpg)



# Calculate the median of hp by cylinder displacement (cyl)
median_hp <- tapply(mtcars$hp, mtcars$cyl, median)
```

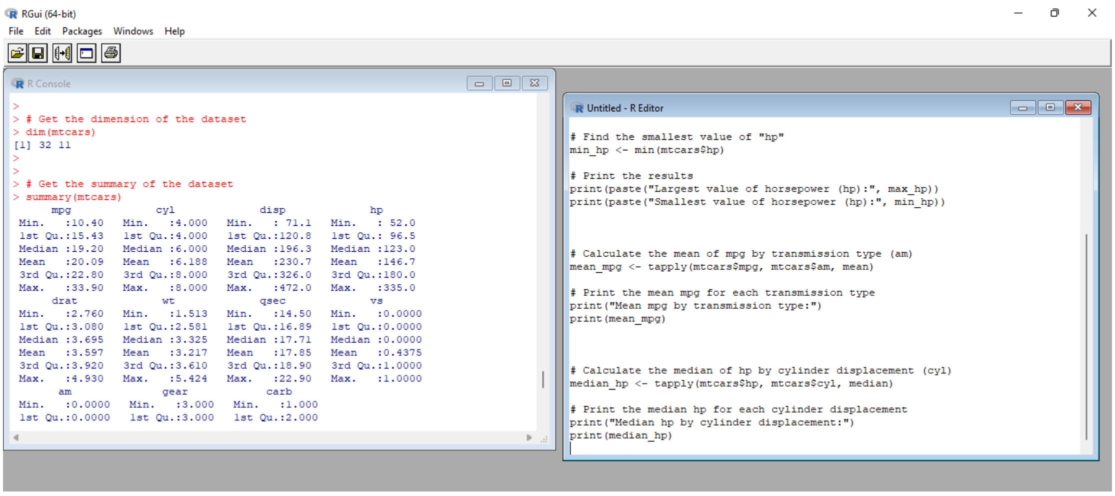
LAB EXERCISE 2

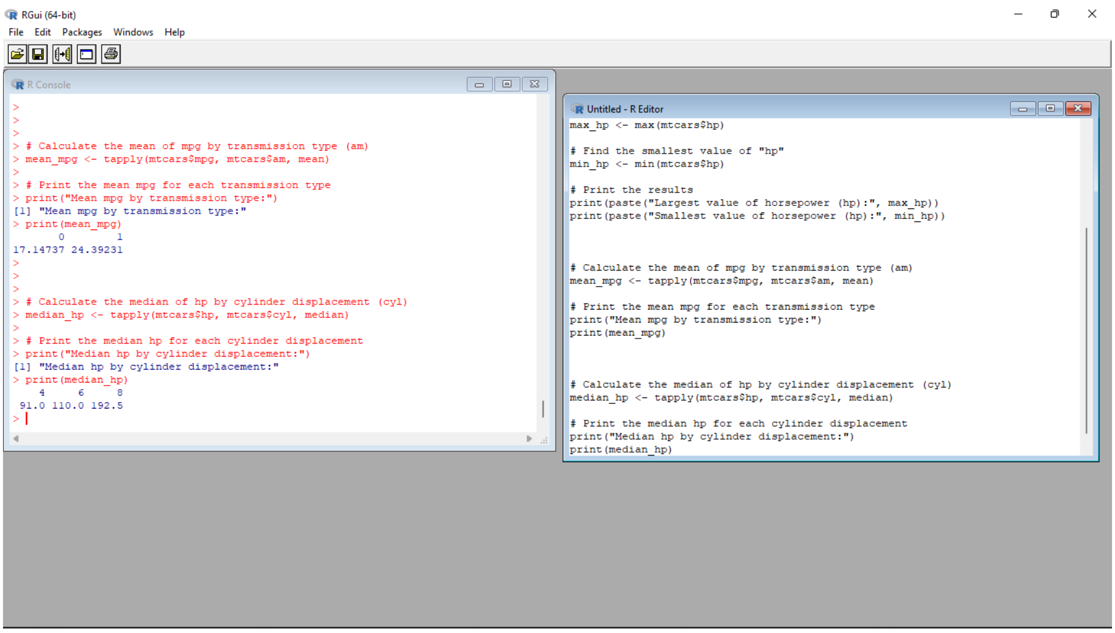
R PROGRAMMING

Print the median hp for each cylinder displacement

print("Median hp by cylinder displacement:")

print(median_hp)





3.(i)Create Scatter plot mpg vs hp, grouped by transmission model (feature named as ‘am’)

(ii)Create Box plot for mpg with respect to transmission model (feature named as ‘am’)

LAB EXERCISE 2

R PROGRAMMING

(iii)Create histogram plot which shows statistical distribution of hp

(iv)Draw the Bar Chart to show car distribution with respect to number of gears grouped by cylinder.(Grouped or multiple bar chart)

(v)Draw Pie chart which shows the percentage of distribution by number of gears.

Input:

```
# Load the mtcars dataset

data(mtcars)


# Scatter plot of mpg vs hp, grouped by transmission model

plot(mtcars$hp, mtcars$mpg, col = mtcars$am, pch = 19, xlab = "Horsepower (hp)", ylab =
"Miles per Gallon (mpg)")

legend("topright", legend = c("Automatic", "Manual"), col = c(1, 2), pch = 19, title =
"Transmission")


# Box plot of mpg with respect to transmission model

boxplot(mpg ~ am, data = mtcars, xlab = "Transmission", ylab = "Miles per Gallon (mpg)",
main = "Box Plot of MPG by Transmission")


# Histogram plot of hp

hist(mtcars$hp, breaks = 10, col = "skyblue", xlab = "Horsepower (hp)", ylab = "Frequency",
main = "Histogram of HP")
```

LAB EXERCISE 2

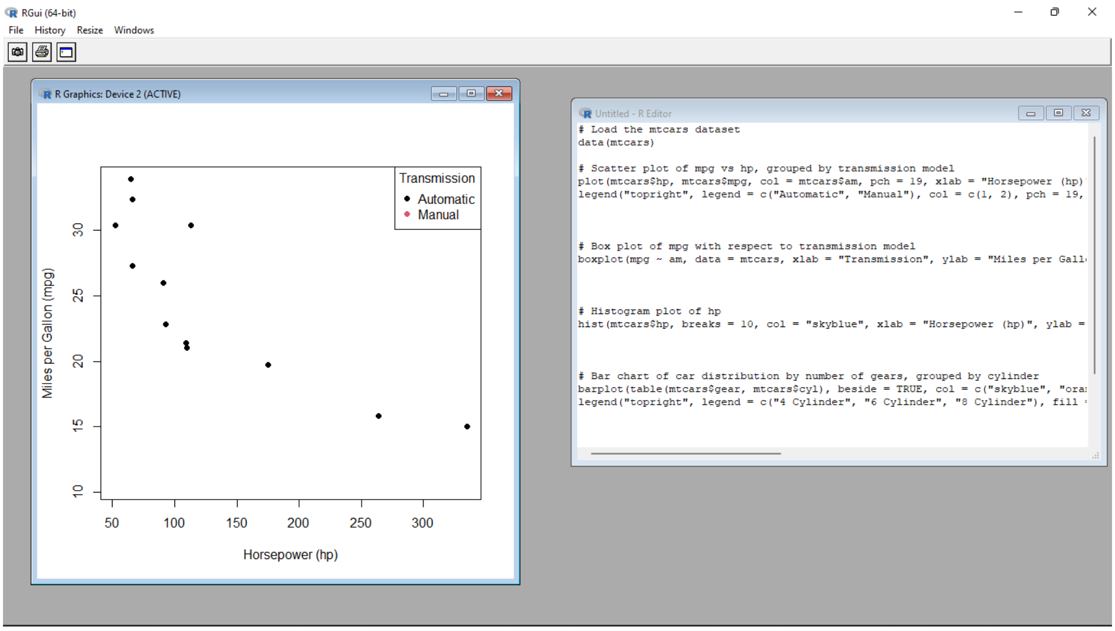
R PROGRAMMING

```
# Bar chart of car distribution by number of gears, grouped by cylinder

barplot(table(mtcars$gear, mtcars$cyl), beside = TRUE, col = c("skyblue", "orange", "green"), xlab = "Number of Gears", ylab = "Frequency", main = "Car Distribution by Number of Gears and Cylinder")

legend("topright", legend = c("4 Cylinder", "6 Cylinder", "8 Cylinder"), fill = c("skyblue", "orange", "green"))
```

```
# Pie chart of percentage distribution by number of gears

gear_counts <- table(mtcars$gear)

labels <- paste(names(gear_counts), "Gears")

percentages <- round(gear_counts / sum(gear_counts) * 100, 1)

pie(gear_counts, labels = labels, main = "Percentage Distribution by Number of Gears")

legend("topright", legend = paste(labels, "-", percentages, "%"), cex = 0.8, fill = rainbow(length(gear_counts)))
```

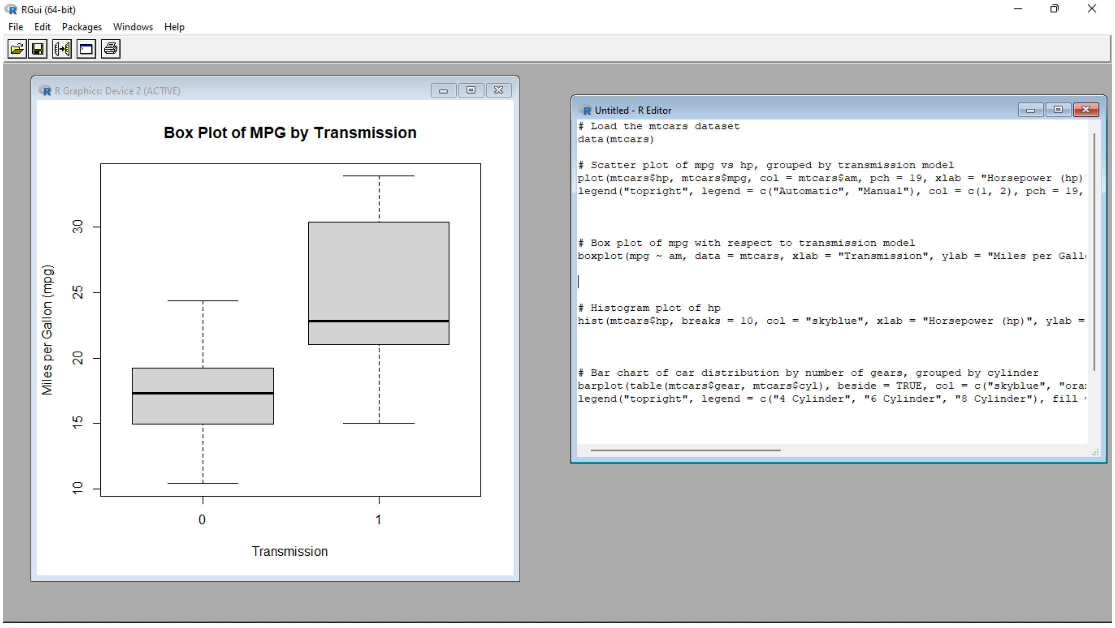
LAB EXERCISE 2

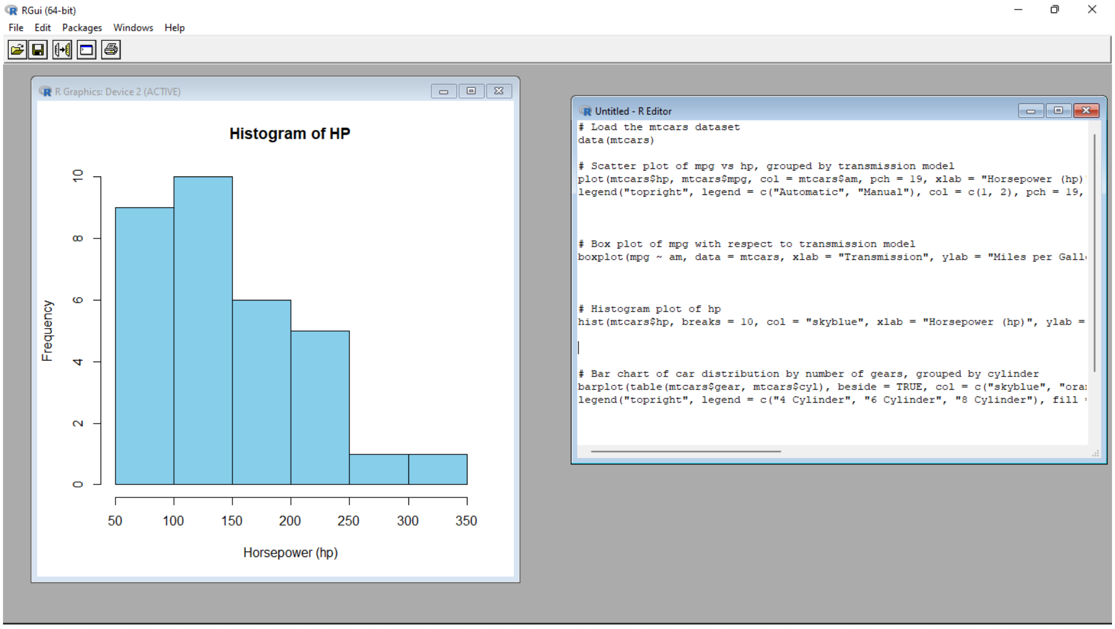
R PROGRAMMING



LAB EXERCISE 2

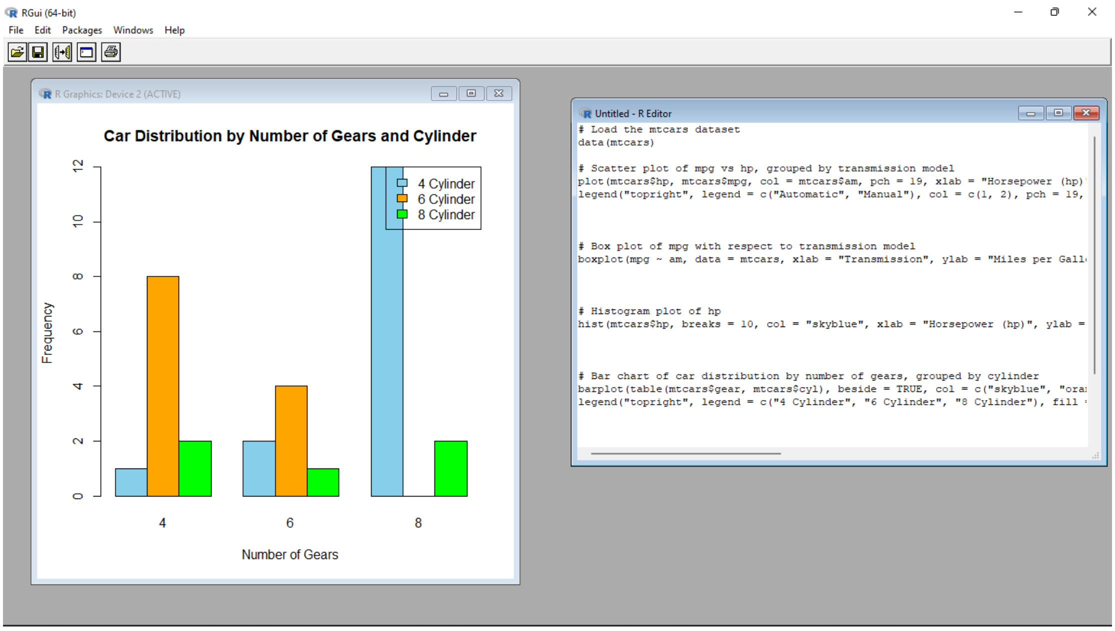
R PROGRAMMING





LAB EXERCISE 2

R PROGRAMMING



4. (i)Generate a multiple regression model using the built-in dataset mtcars. Establish the relationship

between "mpg" as a response variable with "disp","hp" and "wt" as predictor variables .

(ii)Plot the multiple regression line model with above model parameters.

(iii) Predict the mileage of the car with dsp=221, hp=102 and wt=2.91

Input:

Load the mtcars dataset

data(mtcars)

Create the multiple regression model

reg_model <- lm(mpg ~ disp + hp + wt, data = mtcars)

LAB EXERCISE 2

R PROGRAMMING

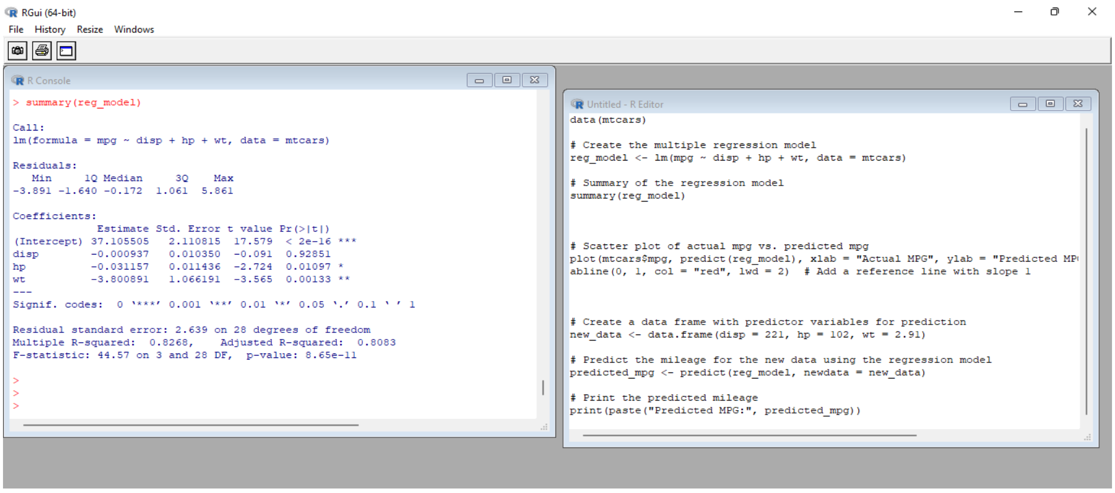
```
# Summary of the regression model
summary(reg_model)
```

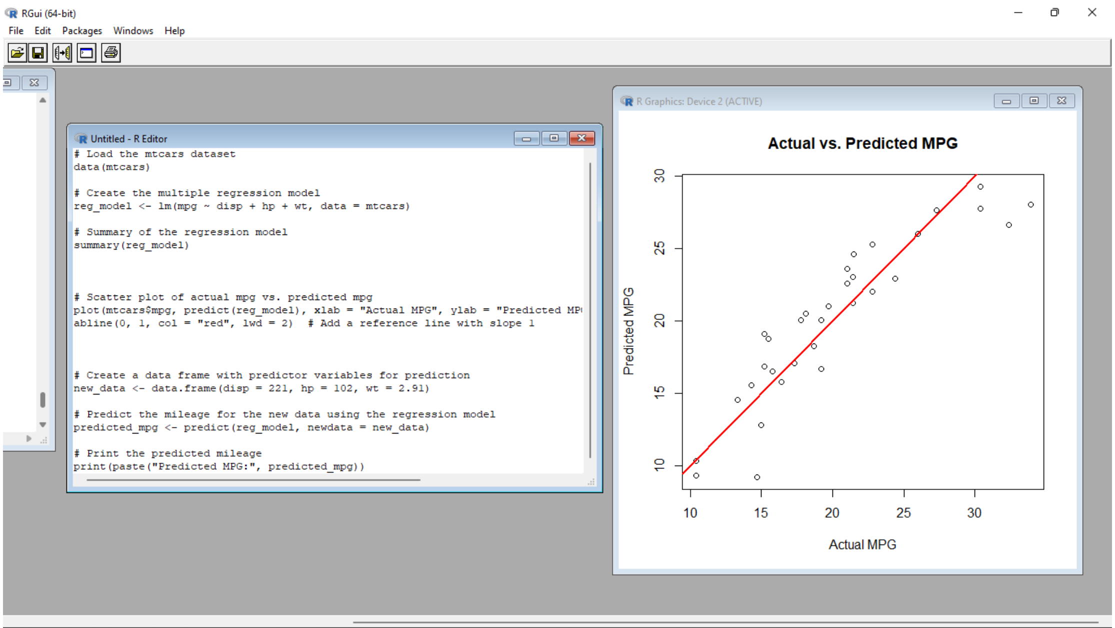
```
# Scatter plot of actual mpg vs. predicted mpg
plot(mtcars$mpg, predict(reg_model), xlab = "Actual MPG", ylab = "Predicted MPG", main
= "Actual vs. Predicted MPG")
abline(0, 1, col = "red", lwd = 2)  # Add a reference line with slope 1
```

```
# Create a data frame with predictor variables for prediction
new_data <- data.frame(disp = 221, hp = 102, wt = 2.91)

# Predict the mileage for the new data using the regression model
predicted_mpg <- predict(reg_model, newdata = new_data)

# Print the predicted mileage
print(paste("Predicted MPG:", predicted_mpg))
```

LAB EXERCISE 2

R PROGRAMMING





Set IV

1. (i) Write a function in R programming to print generate Fibonacci sequence using

Recursion in R .

(ii) Find sum of natural numbers up-to 10, without formula using loop

statement.

(iii) create a vector 1:10 and Find a square of each number and store that in a separate list.

LAB EXERCISE 2

R PROGRAMMING

Input:

i.)
```
fibonacci <- function(n) {
  if (n <= 0) {
    stop("Input must be a positive integer.")
  } else if (n == 1) {
    return(0)
  } else if (n == 2) {
    return(1)
  } else {
    return(fibonacci(n - 1) + fibonacci(n - 2))
  }
}

# Example usage
n <- 10
fib_sequence <- sapply(1:n, fibonacci)
print(fib_sequence)
```

ii.)
```
sum_natural_numbers <- 0

for (i in 1:10) {
  sum_natural_numbers <- sum_natural_numbers + i
}
```

LAB EXERCISE 2

R PROGRAMMING

print(sum_natural_numbers)

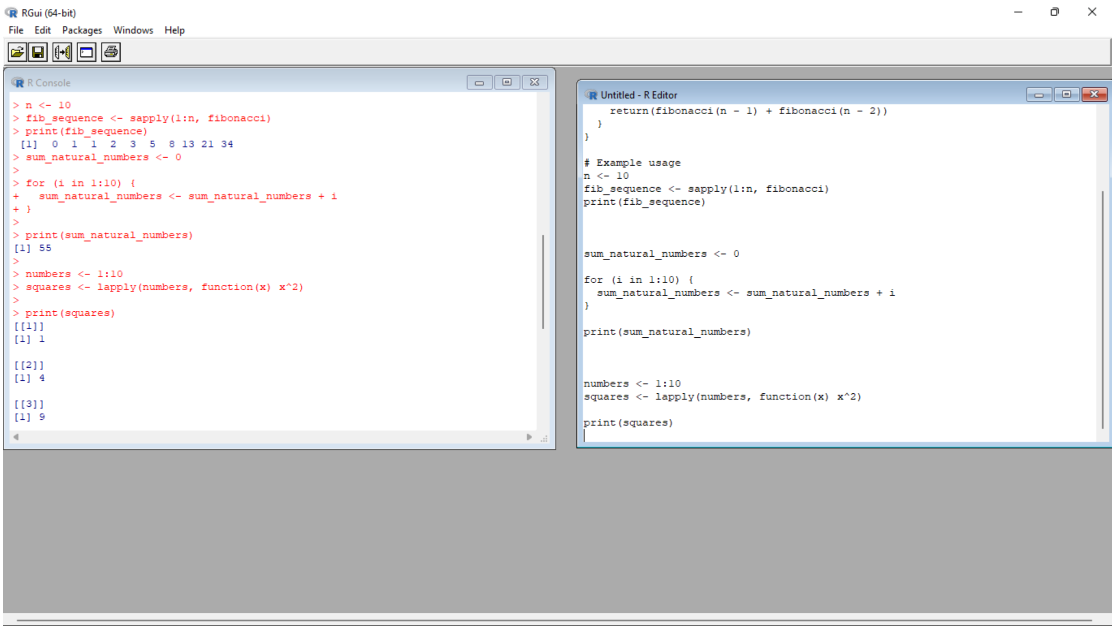
iii.)numbers <- 1:10

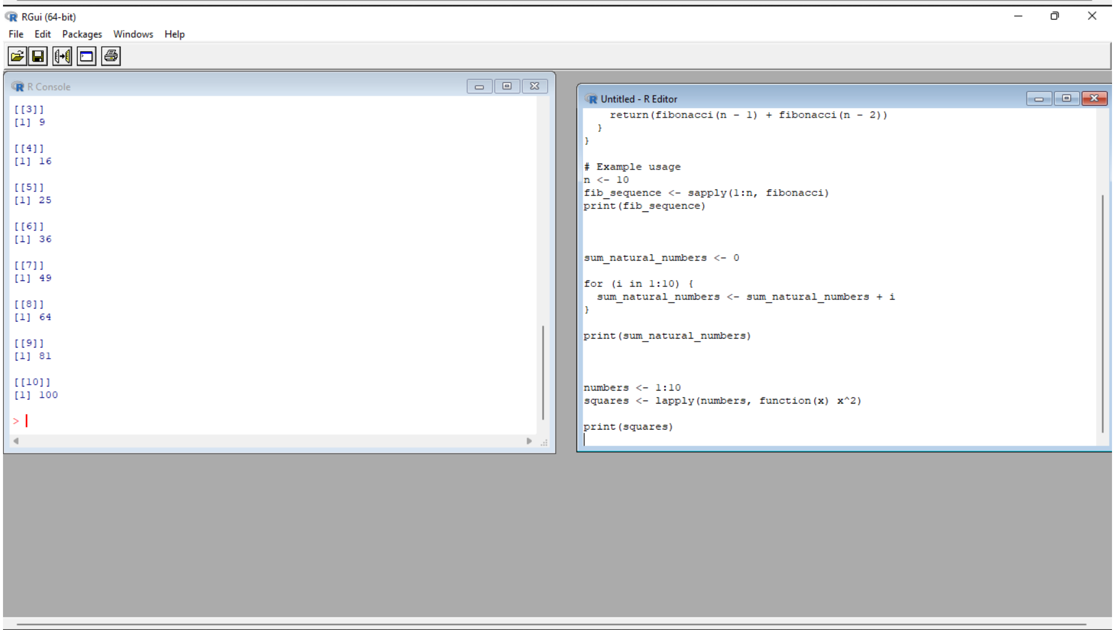
squares <- lapply(numbers, function(x) x^2)

print(squares)

LAB EXERCISE 2

R PROGRAMMING





2. **MTCARS**(motor trend car road test) comprises fuel consumption, performance and

10 aspects of automobile design for 32 automobiles. It comes pre-installed with **dplyr** package

in R.

(i)Find the dimension of the data set

(ii)Give the statistical summary of the features.

LAB EXERCISE 2

R PROGRAMMING

(iii)Print the categorical features in Dataset

(iv)Find the average weight(wt) grouped by Engine shape(vs)

(v)Find the largest and smallest value of the variable weight with respect to Engine shape

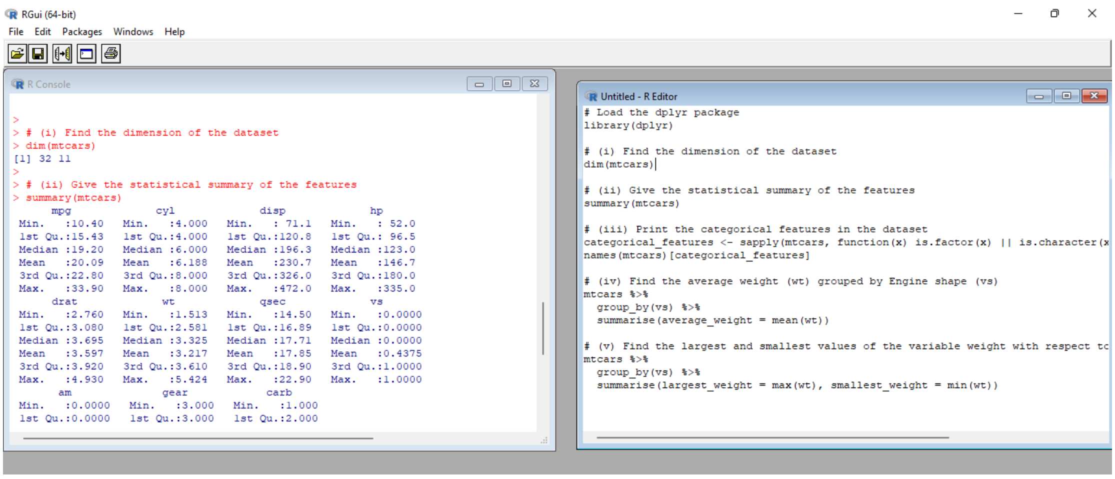
Input:

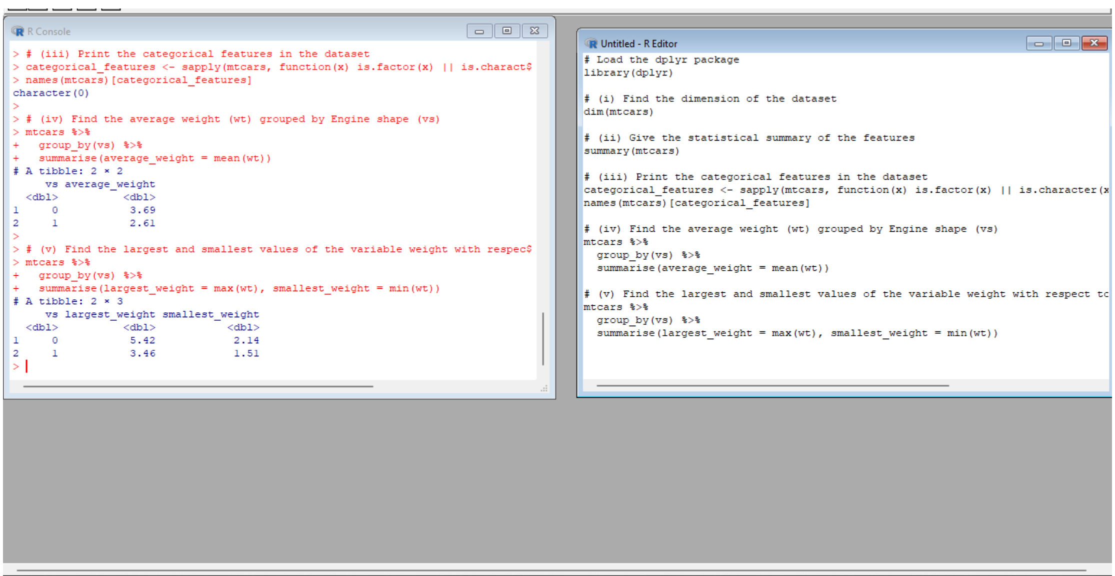
```
# Load the dplyr package
library(dplyr)

# Check the dimension of the mtcars dataset
dim(mtcars)

# Summarize the features of the mtcars dataset
summary(mtcars)


# Identify the categorical features in the mtcars dataset
categorical_features <- sapply(mtcars, is.factor)

# Print the categorical features
names(mtcars)[categorical_features]


# Calculate the average weight (wt) grouped by Engine shape (vs)
mtcars %>%
  group_by(vs) %>%
```

LAB EXERCISE 2

R PROGRAMMING

```
  summarize(avg_weight = mean(wt))


# Find the largest and smallest values of the variable weight with respect to Engine shape

mtcars %>%

  group_by(vs) %>%

  summarize(largest_weight = max(wt),

            smallest_weight = min(wt))
```





LAB EXERCISE 2

R PROGRAMMING

3.Use ggplot package to plot below EDA questions label the plot accordingly

(i)Create weight(wt) vs displacement(disp) scatter plot factor by Engine Shape(vs)

(ii) Create horsepower (hp) vs mileage (mgp) scatter plot factor by Engine Shape(vs)

(iv)In above(ii) plot , Separate columns according to cylinders(cyl) size

(v) Create histogram plot for horsepower (hp) with bin-width size of 5

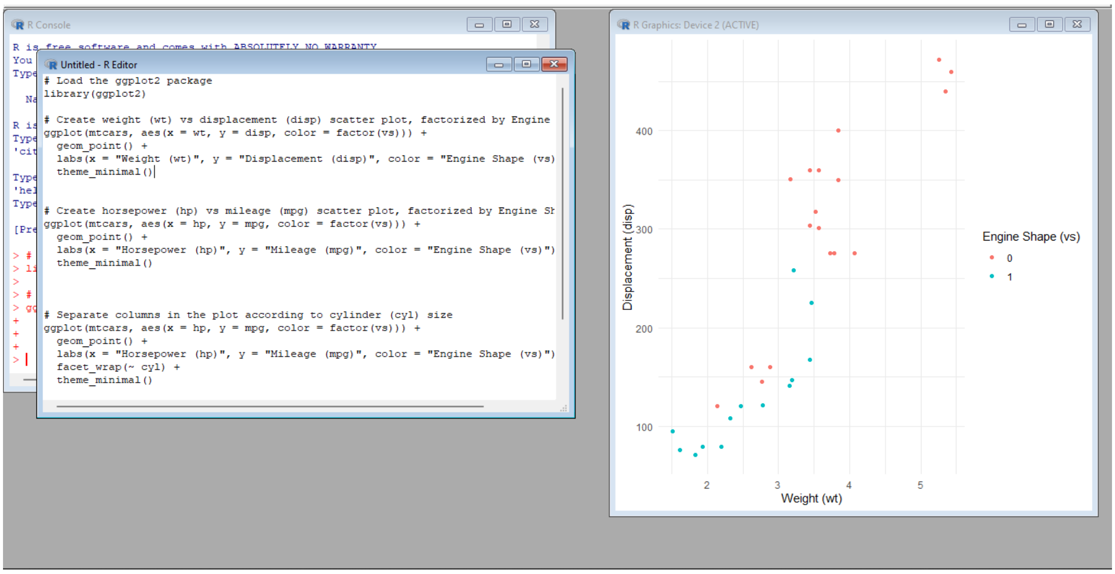
Input:

```
# Load the ggplot2 package
library(ggplot2)

# Create weight (wt) vs displacement (disp) scatter plot, factorized by Engine Shape (vs)
ggplot(mtcars, aes(x = wt, y = disp, color = factor(vs))) +
  geom_point() +
  labs(x = "Weight (wt)", y = "Displacement (disp)", color = "Engine Shape (vs)") +
  theme_minimal()


# Create horsepower (hp) vs mileage (mpg) scatter plot, factorized by Engine Shape (vs)
ggplot(mtcars, aes(x = hp, y = mpg, color = factor(vs))) +
```

```
  geom_point() +
  labs(x = "Horsepower (hp)", y = "Mileage (mpg)", color = "Engine Shape
(vs)") +
  theme_minimal()



# Separate columns in the plot according to cylinder (cyl) size
ggplot(mtcars, aes(x = hp, y = mpg, color = factor(vs))) +
  geom_point() +
  labs(x = "Horsepower (hp)", y = "Mileage (mpg)", color = "Engine Shape
(vs)") +
  facet_wrap(~ cyl) +
  theme_minimal()



# Create a histogram plot for horsepower (hp) with a bin-width size of 5
ggplot(mtcars, aes(x = hp)) +
  geom_histogram(binwidth = 5, fill = "blue", color = "black") +
  labs(x = "Horsepower (hp)", y = "Count") +
  theme_minimal()
```

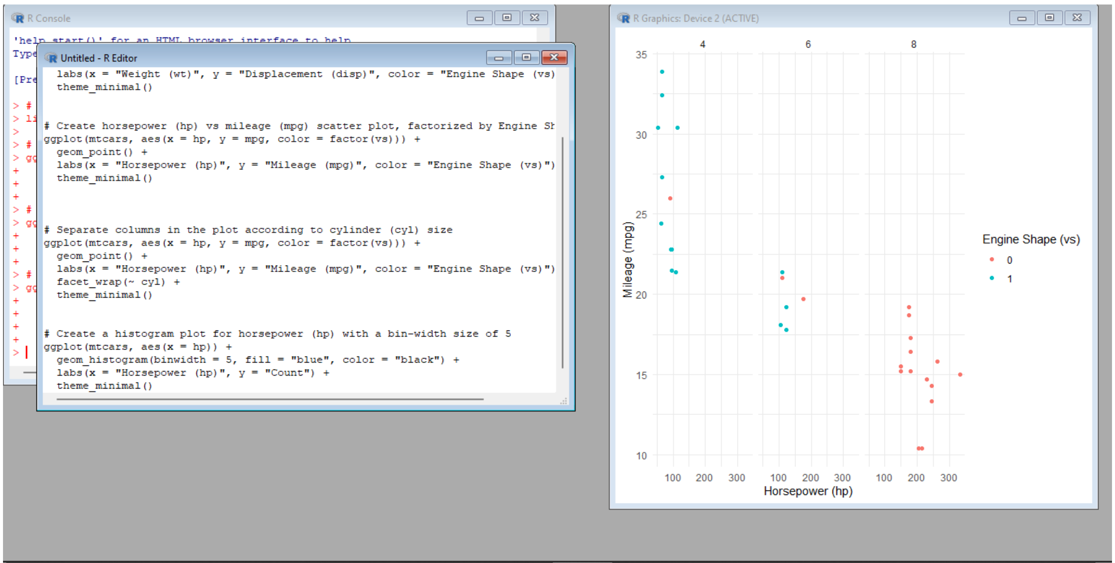
LAB EXERCISE 2

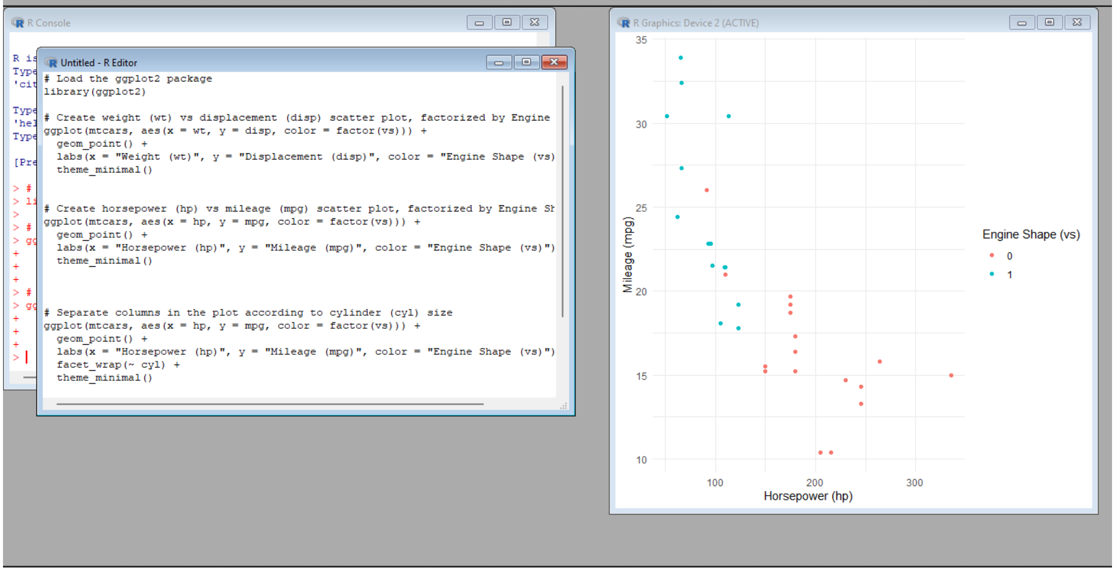
R PROGRAMMING



LAB EXERCISE 2

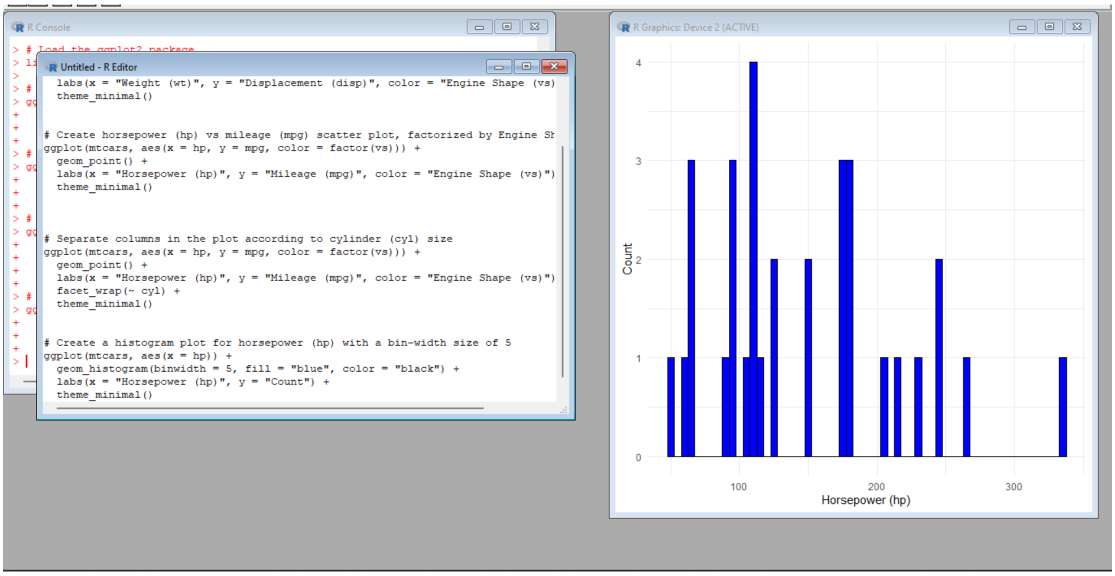
R PROGRAMMING





LAB EXERCISE 2

R PROGRAMMING



4. Performing Logistic regression on dataset to predict the cars Engine shape(vs) .

(i)Do the EDA analysis and find the features which is impact the Engine shape and use this for model.

(ii) Split the data set randomly with 80:20 ration to create train and test dataset and create logistic

model

(iii)Create the Confusion matrix among prediction and test data.

Input:

Load the required packages

library(ggplot2)

library(dplyr)

EDA analysis

Explore the relationship between Engine Shape (vs) and other variables

eda_data <- mtcars %>%

LAB EXERCISE 2

R PROGRAMMING

```
  select(vs, mpg, hp, wt, qsec)  # Include features of interest

# Create scatter plots
scatter_plots <- lapply(names(eda_data)[-1], function(var) {
  ggplot(eda_data, aes_string(x = var, y = "vs")) +
    geom_point() +
    labs(x = var, y = "Engine Shape (vs)") +
    theme_minimal()
})

# Print scatter plots
print(scatter_plots)
```

```
# Load the required package
library(caret)

# Split the dataset into train and test datasets
set.seed(123)
train_indices <- createDataPartition(mtcars$vs, p = 0.8, list = FALSE)
train_data <- mtcars[train_indices, ]
test_data <- mtcars[-train_indices, ]
```

```
# Load the required package
library(caret)
```

LAB EXERCISE 2

R PROGRAMMING

Train the logistic regression model

logistic_model <- train(vs ~ mpg + hp + wt + qsec, data = train_data, method = "glm", family = "binomial")

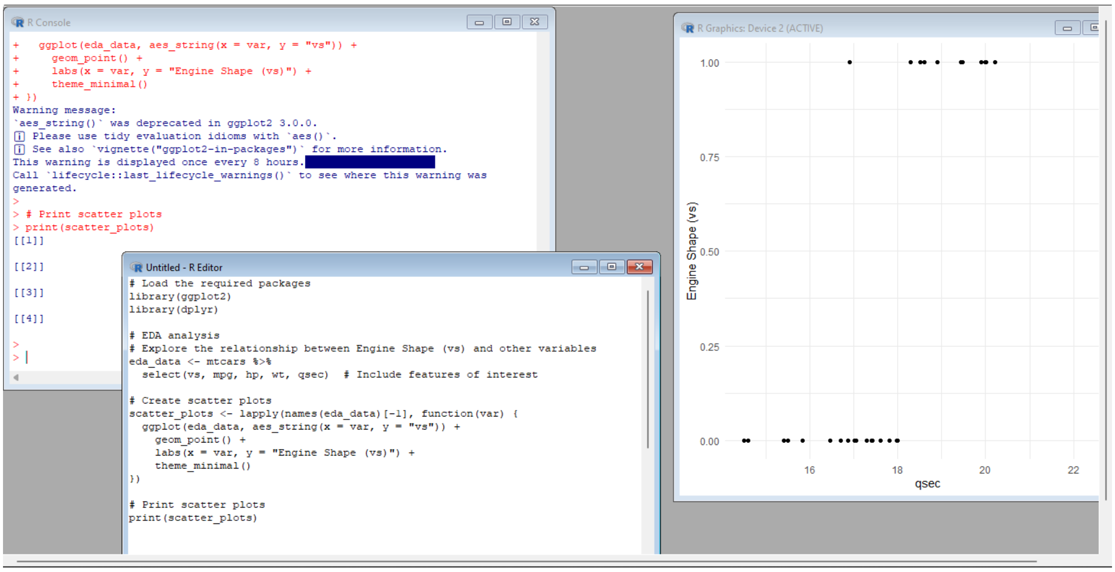
Make predictions on the test dataset

predictions <- predict(logistic_model, newdata = test_data)

Create the confusion matrix

confusion_matrix <- confusionMatrix(predictions, test_data$vs)

print(confusion_matrix)



Set-V

1.(i) Write a R program to extract the five of the levels of factor created from a random sample from the

LETTERS (Part of the base R distribution.)

LAB EXERCISE 2

R PROGRAMMING

(ii)Write R function to find the range of given vector. Range=Max-Min

Sample input, C<-(9,8,7,6,5,4,3,2,1), output=8

(iii)Wirte the R function to find the number of vowels in given string

Sample input c<- “matrix”, output<-2

Input:

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

# Generate a random sample from LETTERS
sample_letters <- sample(LETTERS, 20, replace = TRUE)

# Convert the sample to a factor
sample_factor <- factor(sample_letters)

# Extract five levels from the factor
five_levels <- levels(sample_factor)[1:5]

# Print the five levels
print(five_levels)
```

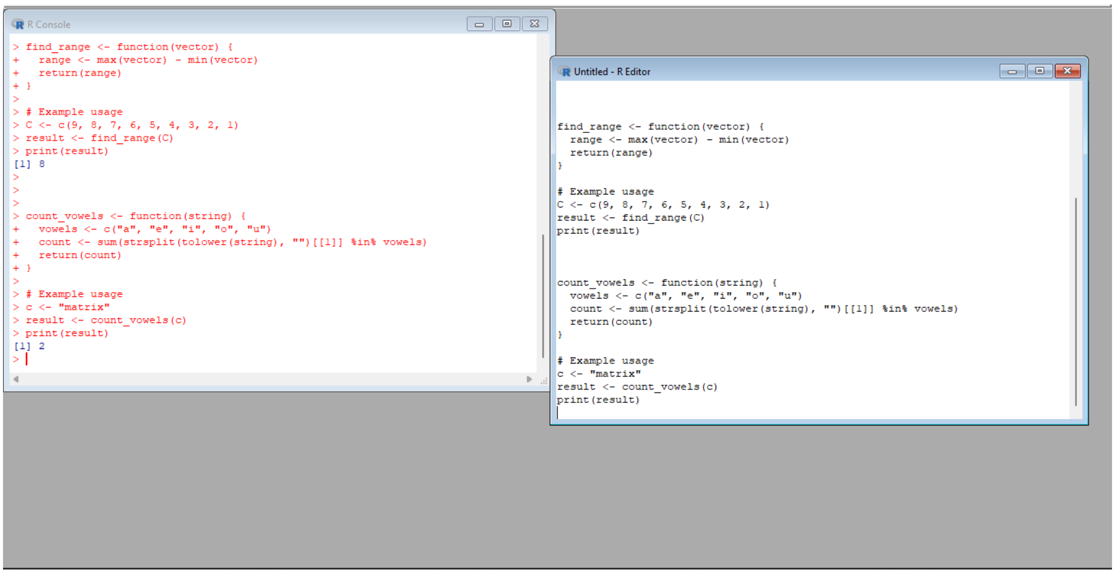
```
ii.)find_range <- function(vector) {
  range <- max(vector) - min(vector)
  return(range)
}
```

LAB EXERCISE 2

R PROGRAMMING

```
# Example usage
C <- c(9, 8, 7, 6, 5, 4, 3, 2, 1)
result <- find_range(C)
print(result)
```

iii.)
```
count_vowels <- function(string) {
  vowels <- c("a", "e", "i", "o", "u")
  count <- sum(strsplit(tolower(string), "")[[1]] %in% vowels)
  return(count)
}

# Example usage
c <- "matrix"
result <- count_vowels(c)
print(result)
```



LAB EXERCISE 2

R PROGRAMMING

2.Load inbuild dataset “ChickWeight” in R

(i) Explore the summary of Data set, like number of Features and its type. Finds the number of records

for each features

(ii)Extract last 6 records of dataset

(iii) order the data frame, in ascending order by feature name “weight” grouped by feature “diet”

(iv)Perform melting function based on “Chick","Time","Diet" features as ID variables

(v)Perform cast function to display the mean value of weight grouped by Diet

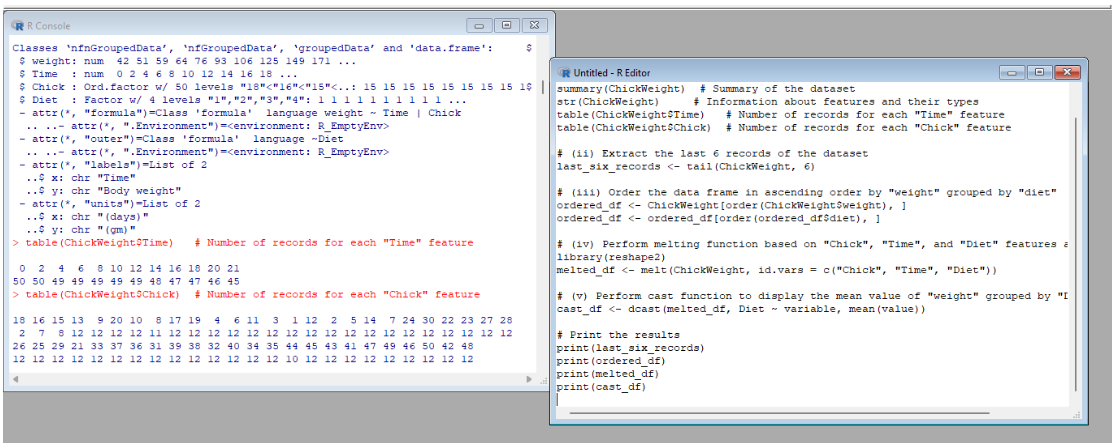
Program :

```
# (i) Load and explore the dataset
data(ChickWeight)
summary(ChickWeight)  # Summary of the dataset
str(ChickWeight)     # Information about features and their types
table(ChickWeight$Time)   # Number of records for each "Time" feature
table(ChickWeight$Chick)  # Number of records for each "Chick" feature

# (ii) Extract the last 6 records of the dataset
last_six_records <- tail(ChickWeight, 6)

# (iii) Order the data frame in ascending order by "weight" grouped by "diet"
ordered_df <- ChickWeight[order(ChickWeight$weight), ]
```

LAB EXERCISE 2

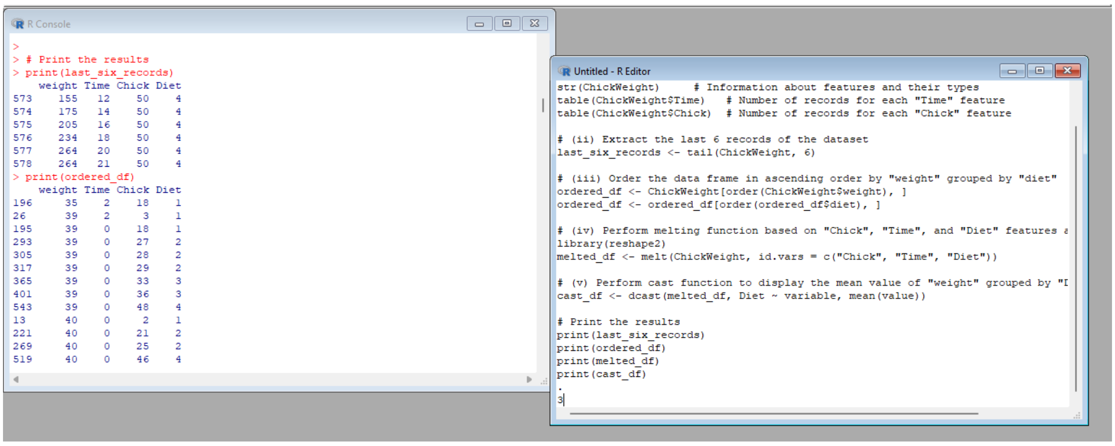
R PROGRAMMING

```
ordered_df <- ordered_df[order(ordered_df$diet), ]


# (iv) Perform melting function based on "Chick", "Time", and "Diet" features as ID variables

library(reshape2)

melted_df <- melt(ChickWeight, id.vars = c("Chick", "Time", "Diet"))


# (v) Perform cast function to display the mean value of "weight" grouped by "Diet"

cast_df <- dcast(melted_df, Diet ~ variable, mean(value))


# Print the results

print(last_six_records)

print(ordered_df)

print(melted_df)

print(cast_df)
```



LAB EXERCISE 2

R PROGRAMMING



3.(i)Get the Statistical Summary of "ChickWeight" dataset

(ii)Create Box plot for "weight" grouped by "Diet"

(iii)Create a Histogram for "Weight" features belong to Diet- 1 category

(iv) Create a Histogram for "Weight" features belong to Diet- 4 category

(v) Create Scatter plot for weight vs Time grouped by Diet

Input :

(i) Get the Statistical Summary of the "ChickWeight" dataset

summary(ChickWeight)

(ii) Create a Box plot for "weight" grouped by "Diet"

boxplot(weight ~ Diet, data = ChickWeight, xlab = "Diet", ylab = "Weight", main = "Weight Distribution by Diet")

(iii) Create a Histogram for "Weight" features belonging to Diet-1 category

LAB EXERCISE 2

R PROGRAMMING

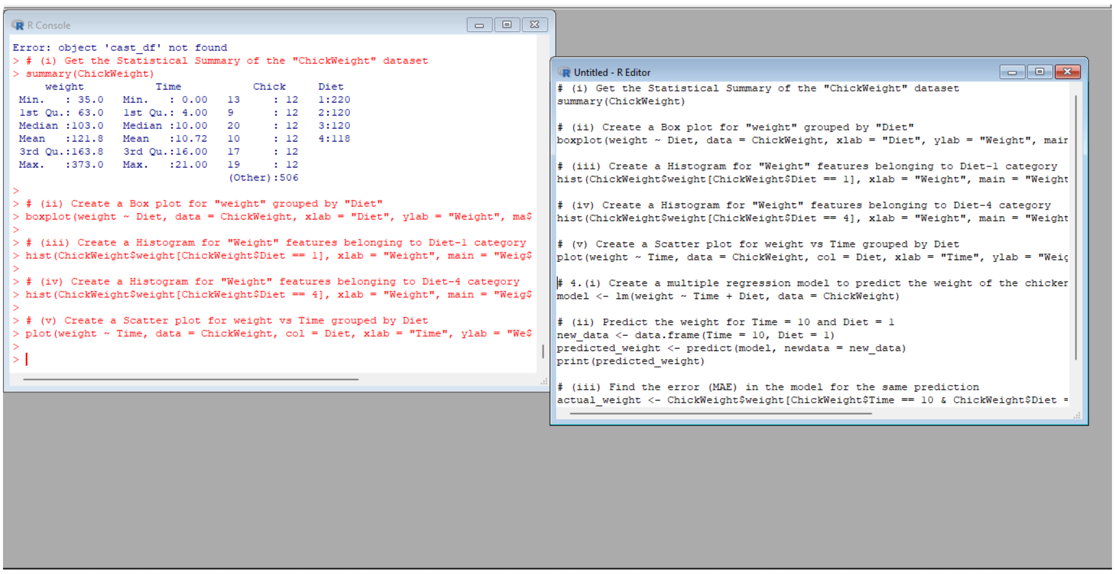
hist(ChickWeight$weight[ChickWeight$Diet == 1], xlab = "Weight", main = "Weight Distribution for Diet-1")

(iv) Create a Histogram for "Weight" features belonging to Diet-4 category

hist(ChickWeight$weight[ChickWeight$Diet == 4], xlab = "Weight", main = "Weight Distribution for Diet-4")

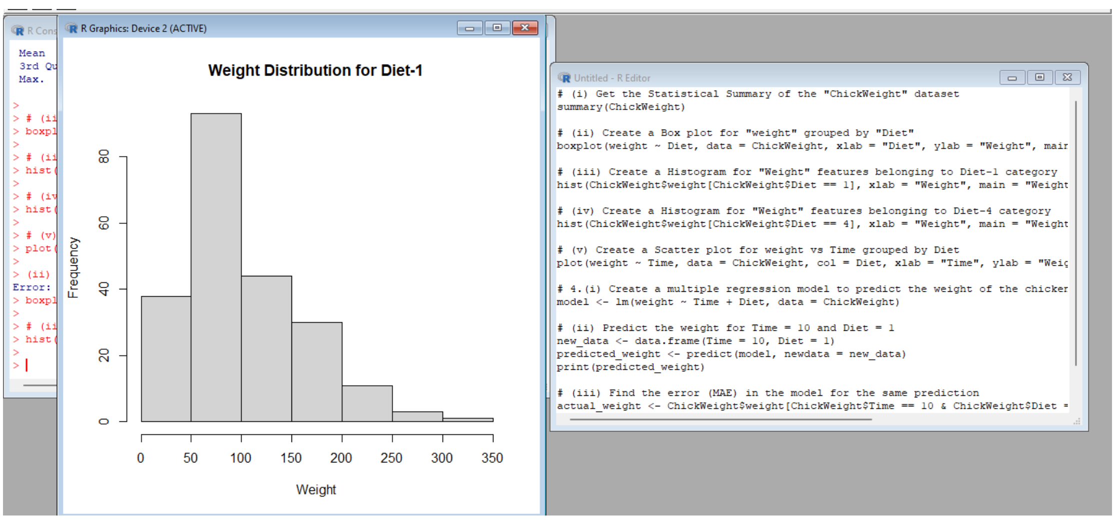
(v) Create a Scatter plot for weight vs Time grouped by Diet

plot(weight ~ Time, data = ChickWeight, col = Diet, xlab = "Time", ylab = "Weight", main = "Weight vs Time by Diet")



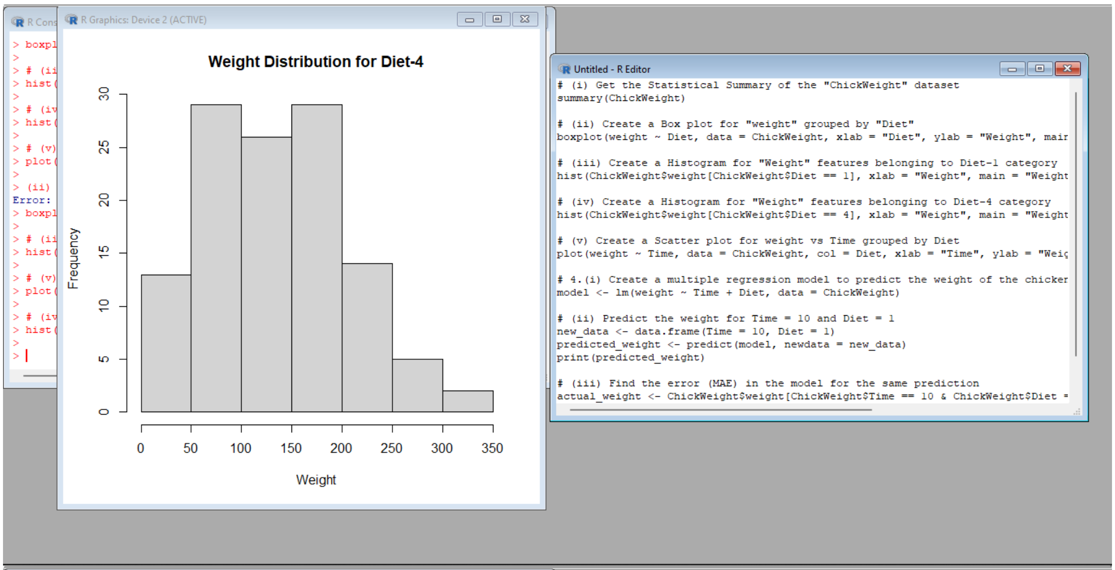
LAB EXERCISE 2

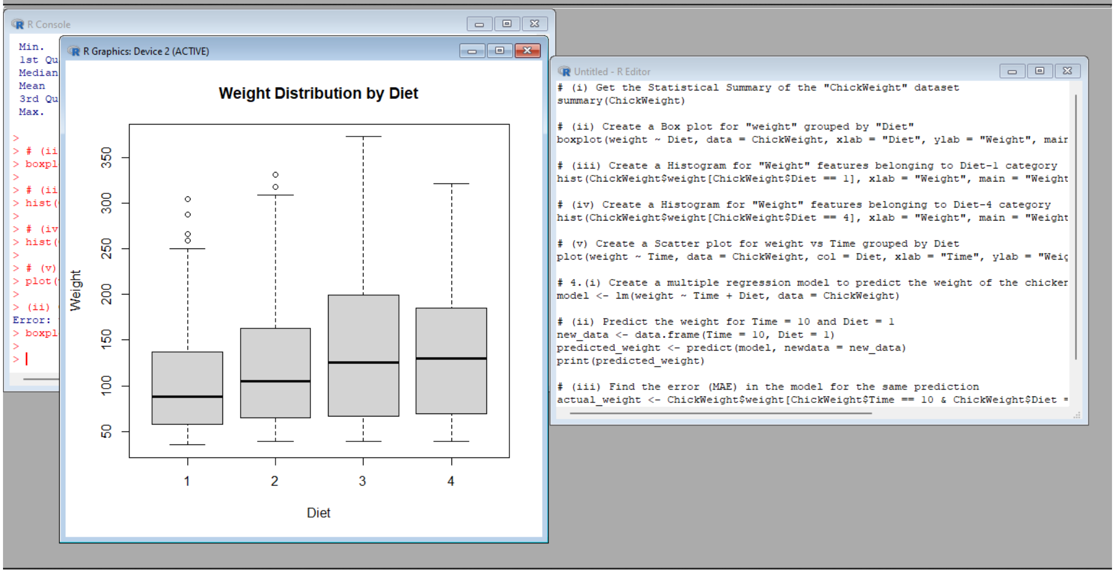
R PROGRAMMING



LAB EXERCISE 2

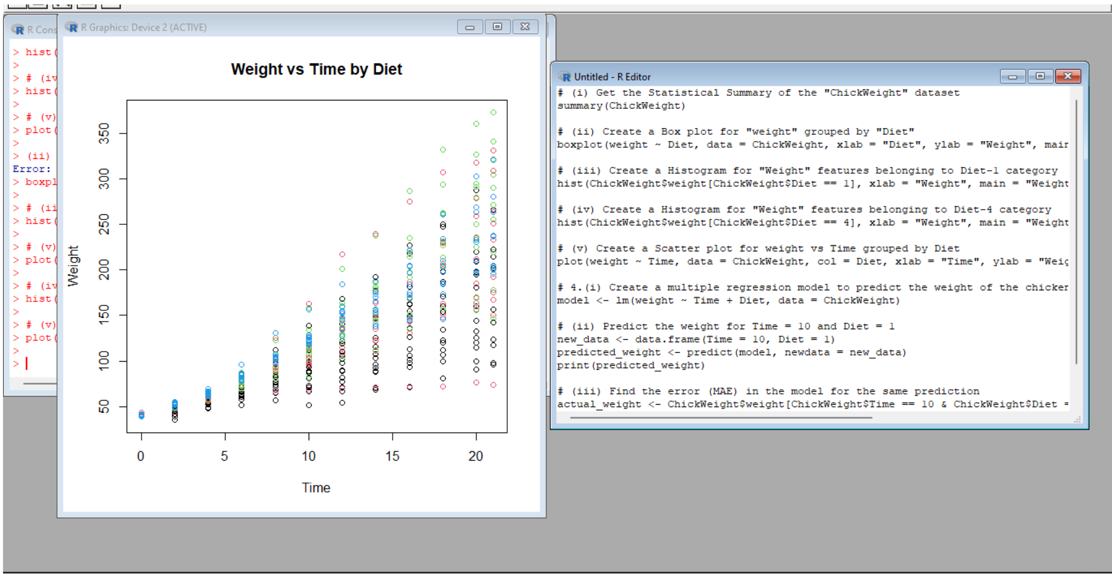
R PROGRAMMING





LAB EXERCISE 2

R PROGRAMMING



V

4.(i) Create multi regression model to find a weight of the chicken , by “Time” and “Diet” as as predictor

variables

(ii) Predict weight for Time=10 and Diet=1

(iii)Find the error(MAE) in model for same

Input :

```
time <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

diet <- c(1, 1, 1, 1, 1, 2, 2, 2, 2, 2)

weight <- c(2.1, 2.4, 2.6, 2.9, 3.2, 3.5, 3.8, 4.1, 4.4, 4.7)

data <- data.frame(time, diet, weight)

model <- lm(weight ~ time + diet, data=data)

new_data <- data.frame(time=10, diet=1)

prediction <- predict(model, newdata=new_data)

cat("Predicted weight: ", prediction, "\n")

actual <- 4.7
```

LAB EXERCISE 2

R PROGRAMMING

```
error <- actual - prediction

cat("Error: ", error, "\n")
```