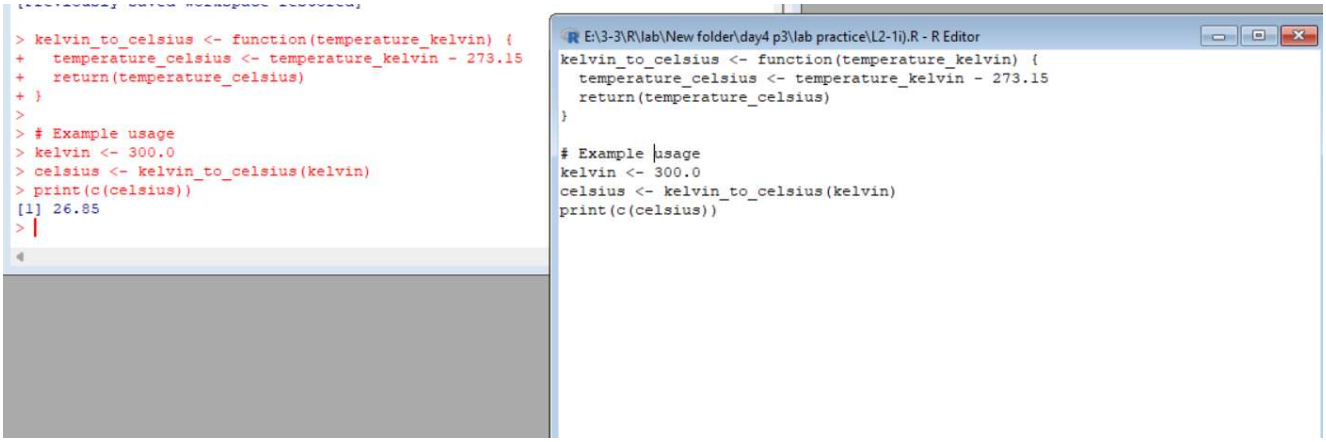
Set-I

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)

SOURCE CODE:

```
kelvin_to_celsius <- function(temperature_kelvin) {
  temperature_celsius <- temperature_kelvin - 273.15
  return(temperature_celsius)
}
# Example usage
kelvin <- 300.0
celsius <- kelvin_to_celsius(kelvin)
print(c(celsius))
```

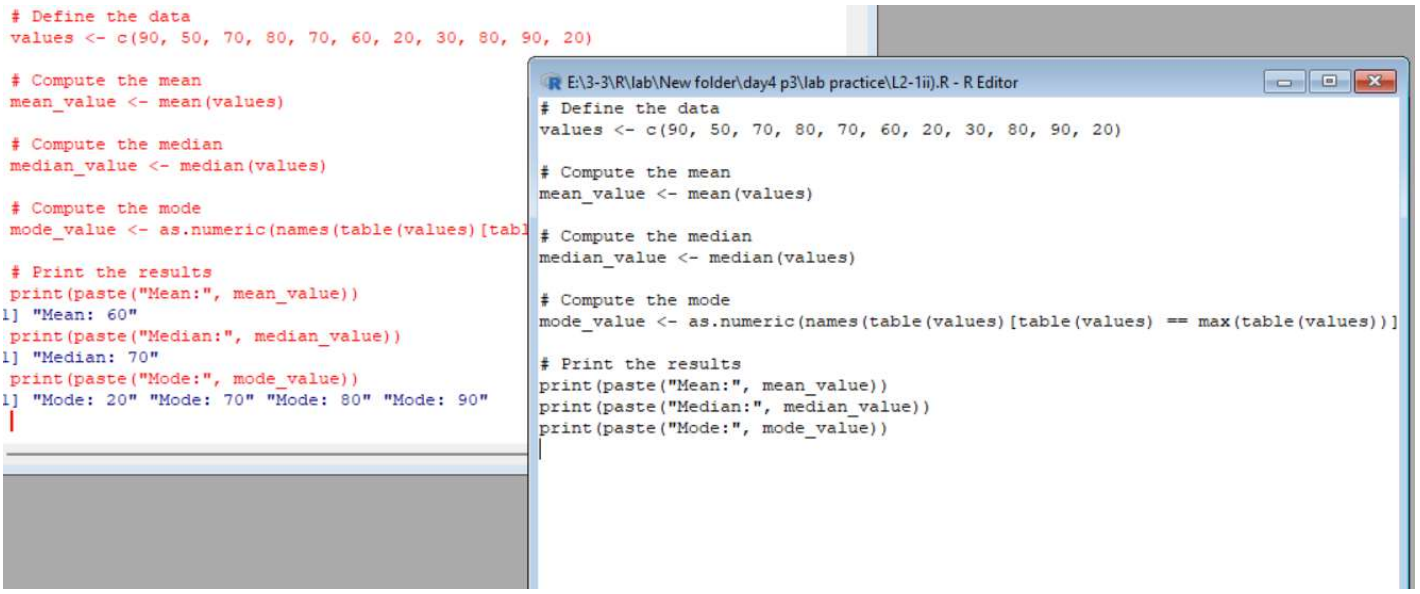


(ii) Write suitable R code to compute the mean, median ,mode of the following values c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)

SOURCE CODE:

```
# Define the data
values <- c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)
# Compute the mean
mean_value <- mean(values)
```

```
# Compute the median

median_value <- median(values)

# Compute the mode

mode_value <- as.numeric(names(table(values)[table(values) == max(table(values))]))

# Print the results

print(paste("Mean:", mean_value))

print(paste("Median:", median_value))

print(paste("Mode:", mode_value))
```

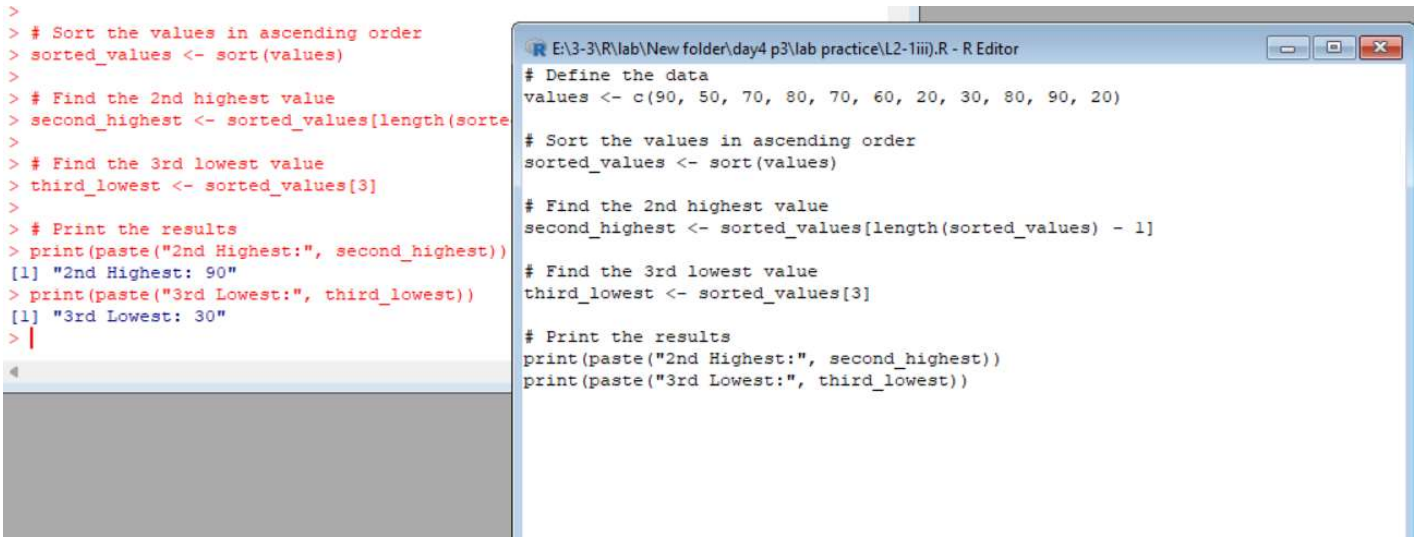


(iii) Write R code to find 2nd highest and 3rd Lowest value of above problem.

SOURCE CODE:

```
# Define the data

values <- c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)

# Sort the values in ascending order

sorted_values <- sort(values)

# Find the 2nd highest value
```

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

# Find the 3rd lowest value

third_lowest <- sorted_values[3]

# Print the results

print(paste("2nd Highest:", second_highest))

print(paste("3rd Lowest:", third_lowest))
```



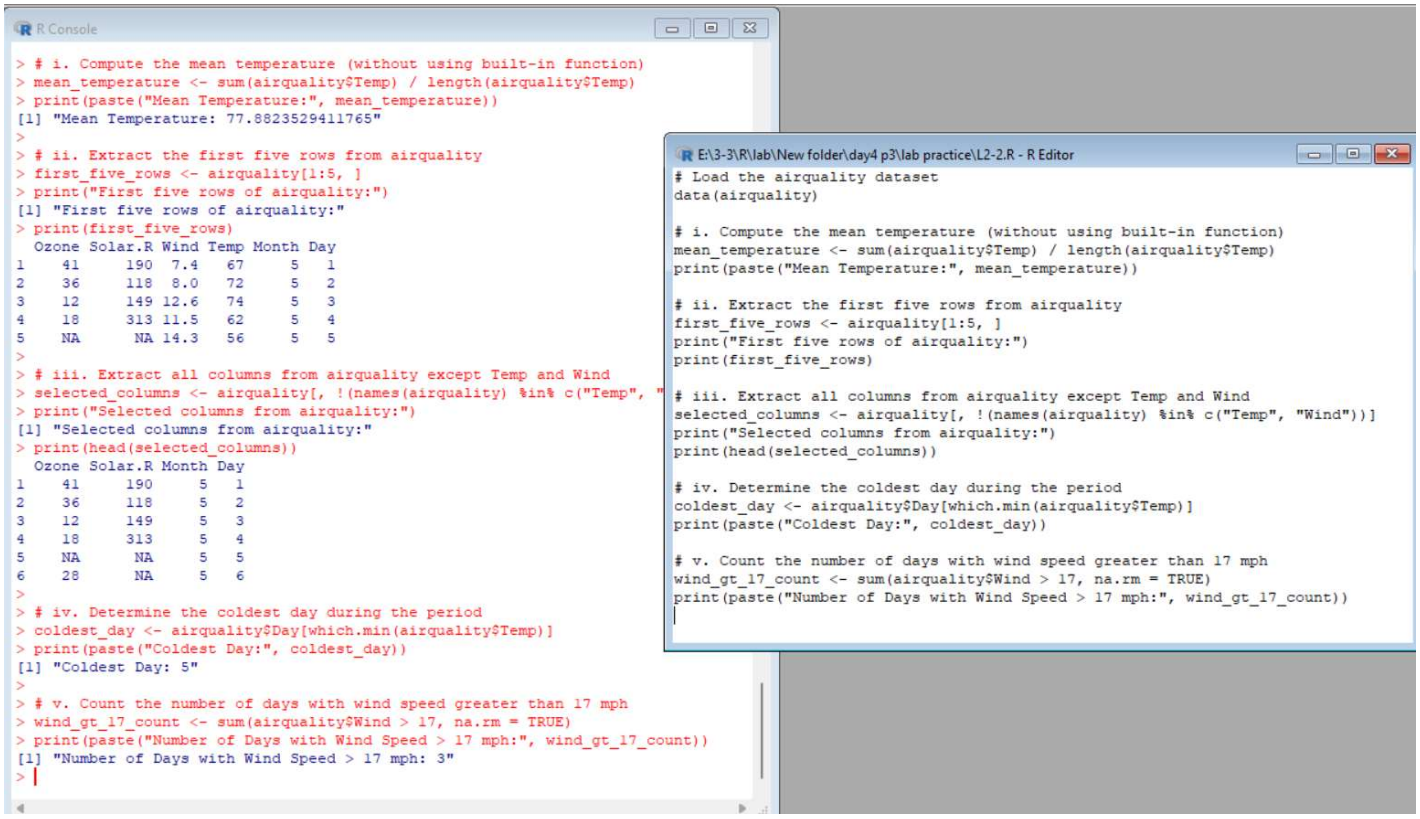
2. Explore the airquality dataset. It contains daily air quality measurements from New York during a period of five months:
   - Ozone: mean ozone concentration (ppb),
   - Solar.R: solar radiation (Langley),
   - Wind: average wind speed (mph),
   - Temp: maximum daily temperature in degrees Fahrenheit,
   - 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) 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?

SOURCE CODE:

```
# Load the airquality dataset
```

```
data(airquality)

# i. Compute the mean temperature (without using built-in function)
mean_temperature <- sum(airquality$Temp) / length(airquality$Temp)
print(paste("Mean Temperature:", mean_temperature))

# ii. Extract the first five rows from airquality
first_five_rows <- airquality[1:5, ]
print("First five rows of airquality:")
print(first_five_rows)

# iii. Extract all columns from airquality except Temp and Wind
selected_columns <- airquality[, !(names(airquality) %in% c("Temp", "Wind"))]
print("Selected columns from airquality:")
print(selected_columns)

# iv. Determine the coldest day during the period
coldest_day <- airquality$Day[which.min(airquality$Temp)]
print(paste("Coldest Day:", coldest_day))

# v. Count the number of days with wind speed greater than 17 mph
wind_gt_17_count <- sum(airquality$Wind > 17, na.rm = TRUE)
print(paste("Number of Days with Wind Speed > 17 mph:", wind_gt_17_count))
```



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"?

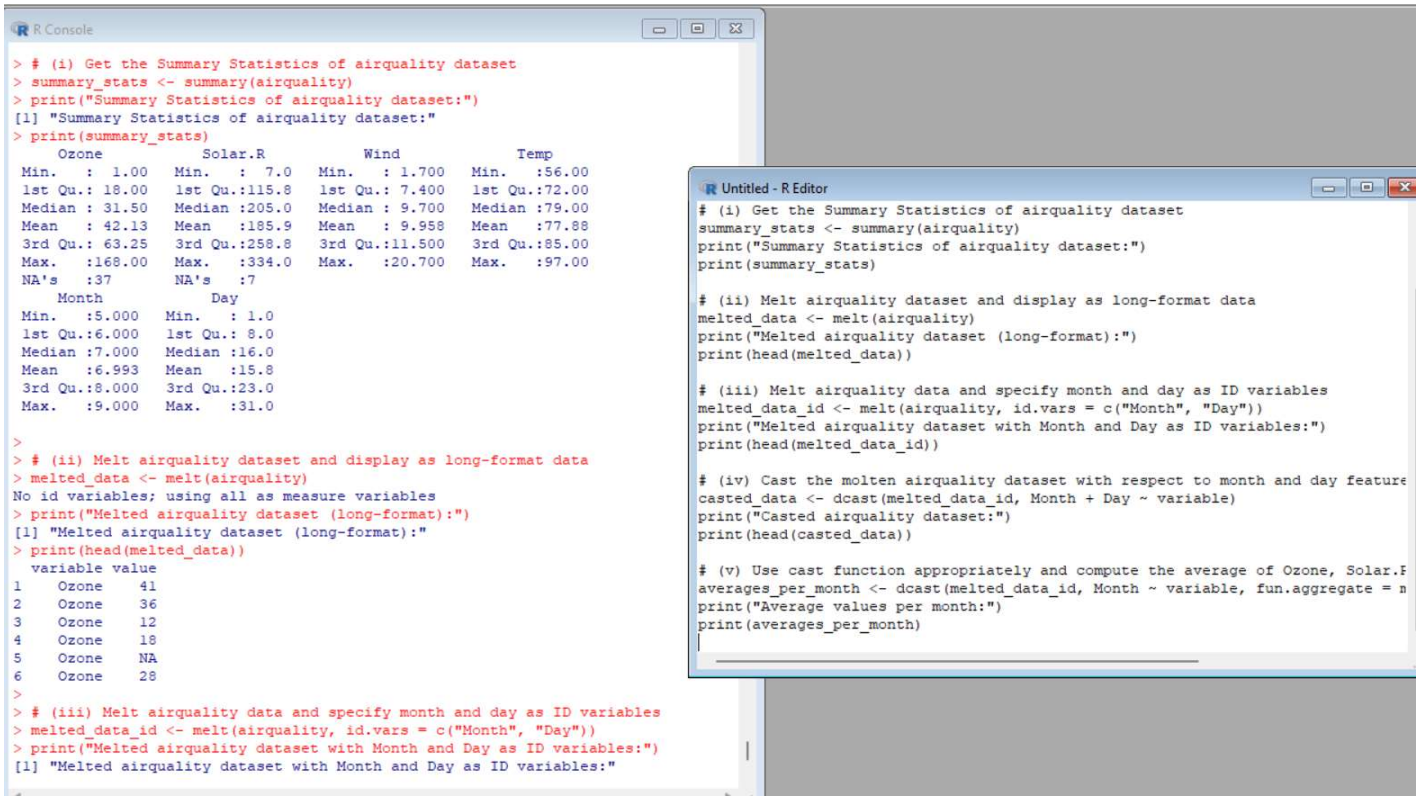
(iv)Cast the molten airquality data set with respect to month and date features

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

SOURCE CODE:

```
# Load the airquality dataset
data(airquality)
library(reshape2)
# (i) Get the Summary Statistics of airquality dataset
summary_stats <- summary(airquality)
print("Summary Statistics of airquality dataset:")
print(summary_stats)

# (ii) Melt airquality dataset and display as long-format data
melted_data <- melt(airquality)
print("Melted airquality dataset (long-format):")
print(head(melted_data))

# (iii) Melt airquality data and specify month and day as ID variables
```

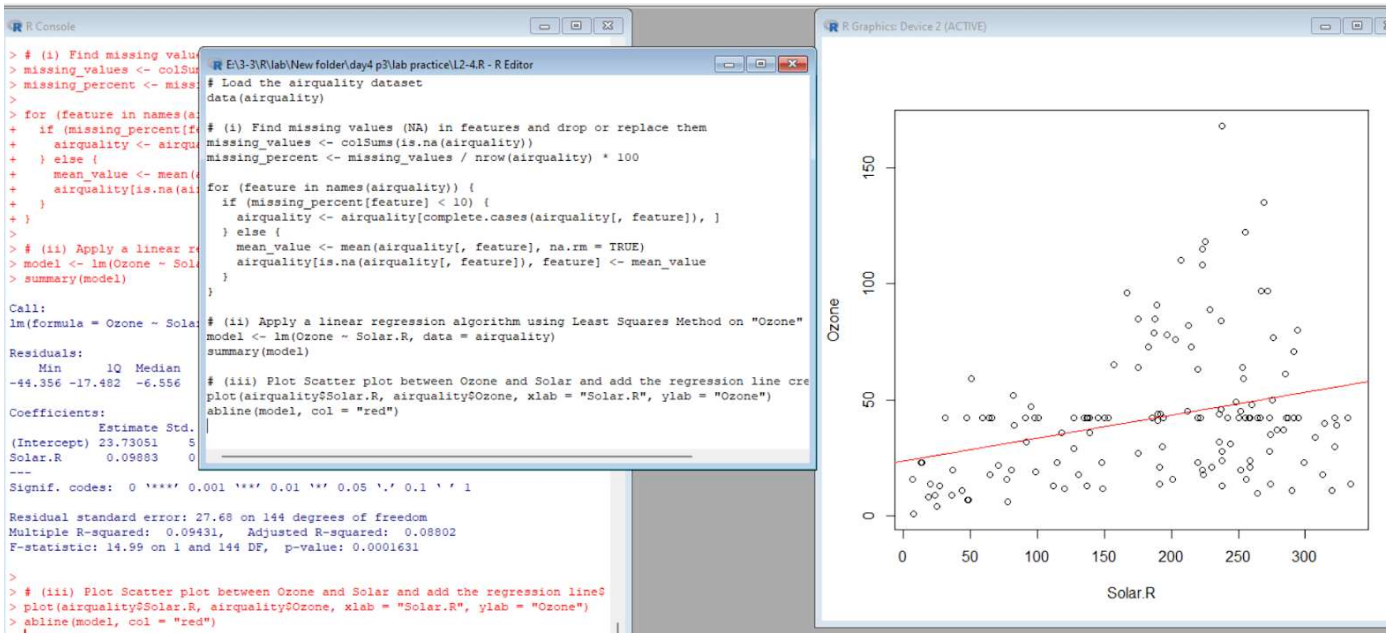
```
melted_data_id <- melt(airquality, id.vars = c("Month", "Day"))
print("Melted airquality dataset with Month and Day as ID variables:")
print(head(melted_data_id))

# (iv) Cast the molten airquality dataset with respect to month and day features
casted_data <- dcast(melted_data_id, Month + Day ~ variable)
print("Casted airquality dataset:")
print(head(casted_data))

# (v) Use cast function appropriately and compute the average of Ozone, Solar.R, Wind, and Temperature per month
averages_per_month <- dcast(melted_data_id, Month ~ variable, fun.aggregate = mean)
print("Average values per month:")
print(averages_per_month)
```



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”

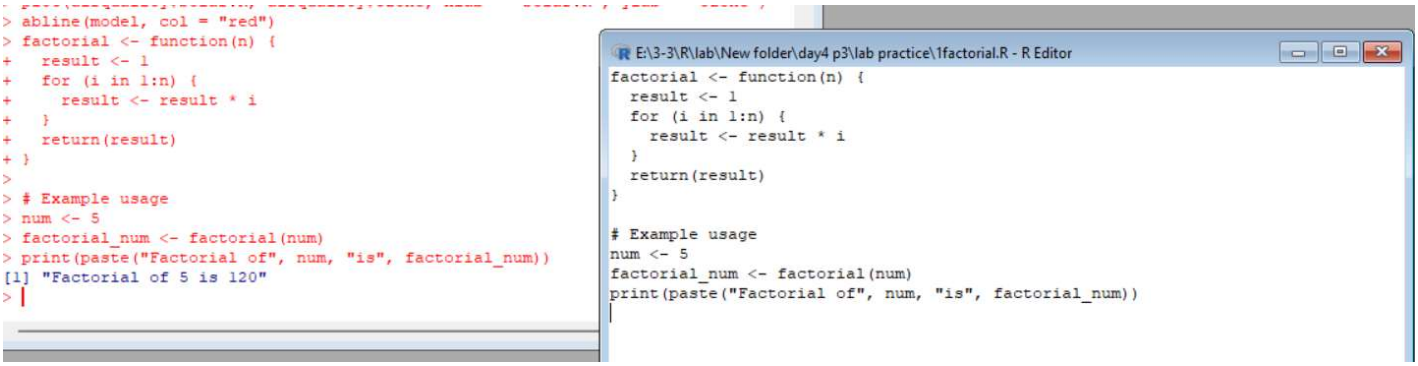
(iii)Plot Scatter plot between Ozone and Solar and add regression line created by above model Set-II



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

SOURCE CODE:

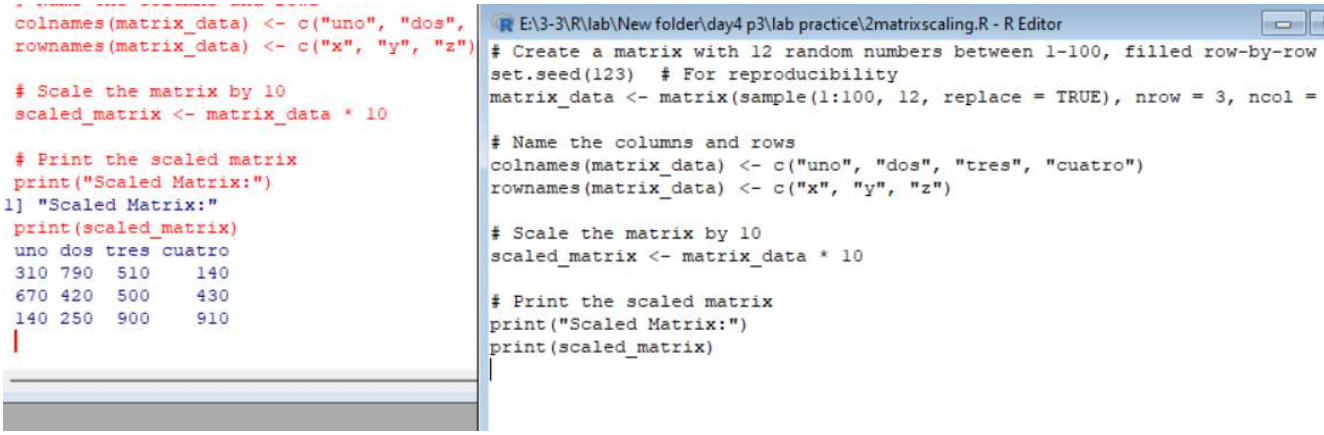
```
factorial <- function(n) {
  result <- 1
  for (i in 1:n) {
    result <- result * i
  }
  return(result)
}

# Example usage
num <- 5
factorial_num <- factorial(num)
print(paste("Factorial of", num, "is", factorial_num))
```



(ii) Create a 3x4 matrix with 12 random numbers between 1-100; have the matrix be filled our rowby-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.

SOURCE CODE:

```
# Create a matrix with 12 random numbers between 1-100, filled row-by-row
set.seed(123)  # For reproducibility
matrix_data <- matrix(sample(1:100, 12, replace = TRUE), nrow = 3, ncol = 4, byrow = TRUE)
# Name the columns and rows
colnames(matrix_data) <- c("uno", "dos", "tres", "cuatro")
rownames(matrix_data) <- c("x", "y", "z")
# Scale the matrix by 10
scaled_matrix <- matrix_data * 10
# Print the scaled matrix
print("Scaled Matrix:")
print(scaled_matrix)
```



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

SOURCE CODE:

```
# Extract the column called "uno" from the original matrix
```

```
column_uno <- matrix_data[, "uno"]

# Print the extracted column as a vector
print("Column 'uno':")
print(column_uno)
```

```
> # Extract the column called "uno" from the ori
> column_uno <- matrix_data[, "uno"]
>
> # Print the extracted column as a vector
> print("Column 'uno':")
[1] "Column 'uno':"
> print(column_uno)
 x  y  z
31 67 14
>
```

E:\3-3\R\lab\New folder\day4 p3\lab practice\3Matrixextract.R - R Editor

```
# Extract the column called "uno" from the original matrix
column_uno <- matrix_data[, "uno"]

# Print the extracted column as a vector
print("Column '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

(iv)create new data frame named iris1 which have a new column name **Sepal.Length.Cate** that categorizes “Sepal.Length” by quantile

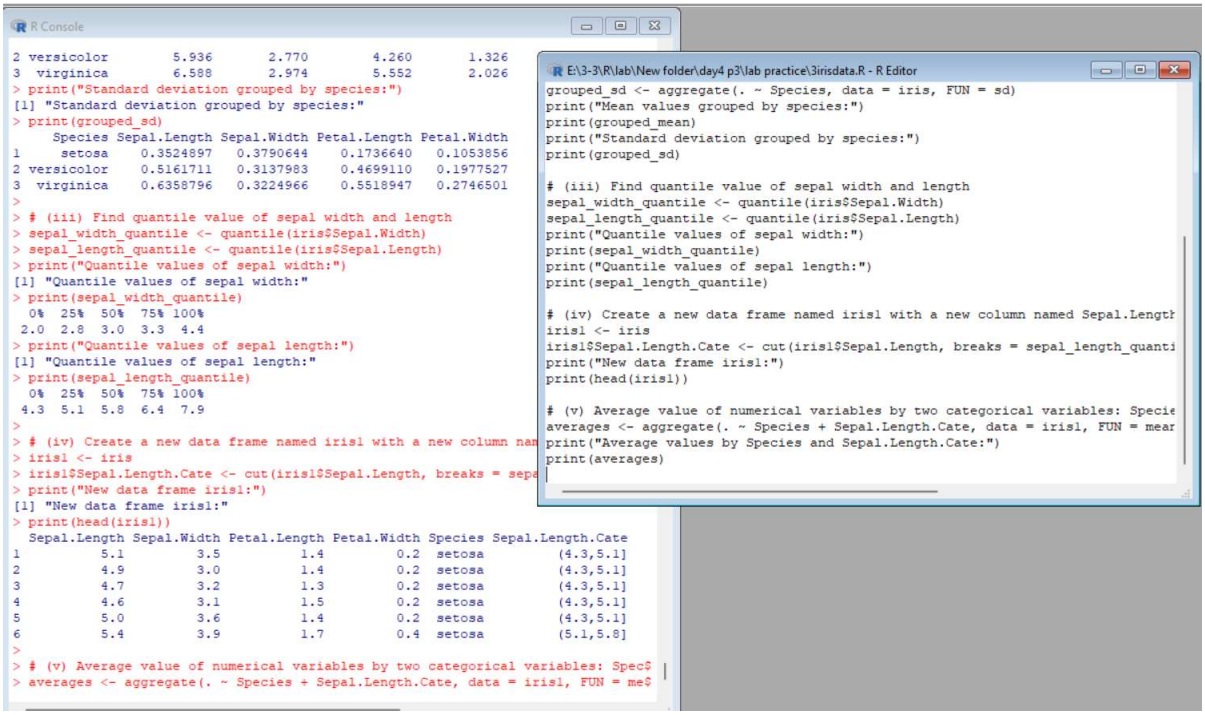
(v) Average value of numerical varialbes by two categorical variables: Species and Sepal.Length.Cate.

SOURCE CODE:

```
# Load the iris dataset

data(iris)

# (i) Find dimension, structure, summary statistics, and standard deviation of all features

print("Dimension:")

print(dim(iris))

print("Structure:")
```

```
print(str(iris))

print("Summary Statistics:")

print(summary(iris))

print("Standard Deviation:")

print(sapply(iris, sd))

# (ii) Find mean and standard deviation of features grouped by three species of Iris flowers

grouped_mean <- aggregate(. ~ Species, data = iris, FUN = mean)

grouped_sd <- aggregate(. ~ Species, data = iris, FUN = sd)

print("Mean values grouped by species:")

print(grouped_mean)

print("Standard deviation grouped by species:")

print(grouped_sd)

# (iii) Find quantile value of sepal width and length

sepal_width_quantile <- quantile(iris$Sepal.Width)

sepal_length_quantile <- quantile(iris$Sepal.Length)

print("Quantile values of sepal width:")

print(sepal_width_quantile)

print("Quantile values of sepal length:")

print(sepal_length_quantile)

# (iv) Create a new data frame named iris1 with a new column named Sepal.Length.Cate that
      categorizes "Sepal.Length" by quantile

iris1 <- iris
```

```
iris1$Sepal.Length.Cate <- cut(iris1$Sepal.Length, breaks = sepal_length_quantile)

print("New data frame iris1:")

print(head(iris1))

# (v) Average value of numerical variables by two categorical variables: Species and
      Sepal.Length.Cate

averages <- aggregate(. ~ Species + Sepal.Length.Cate, data = iris1, FUN = mean)

print("Average values by Species and Sepal.Length.Cate:")

print(averages)
```



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
   (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

SOURCE CODE:

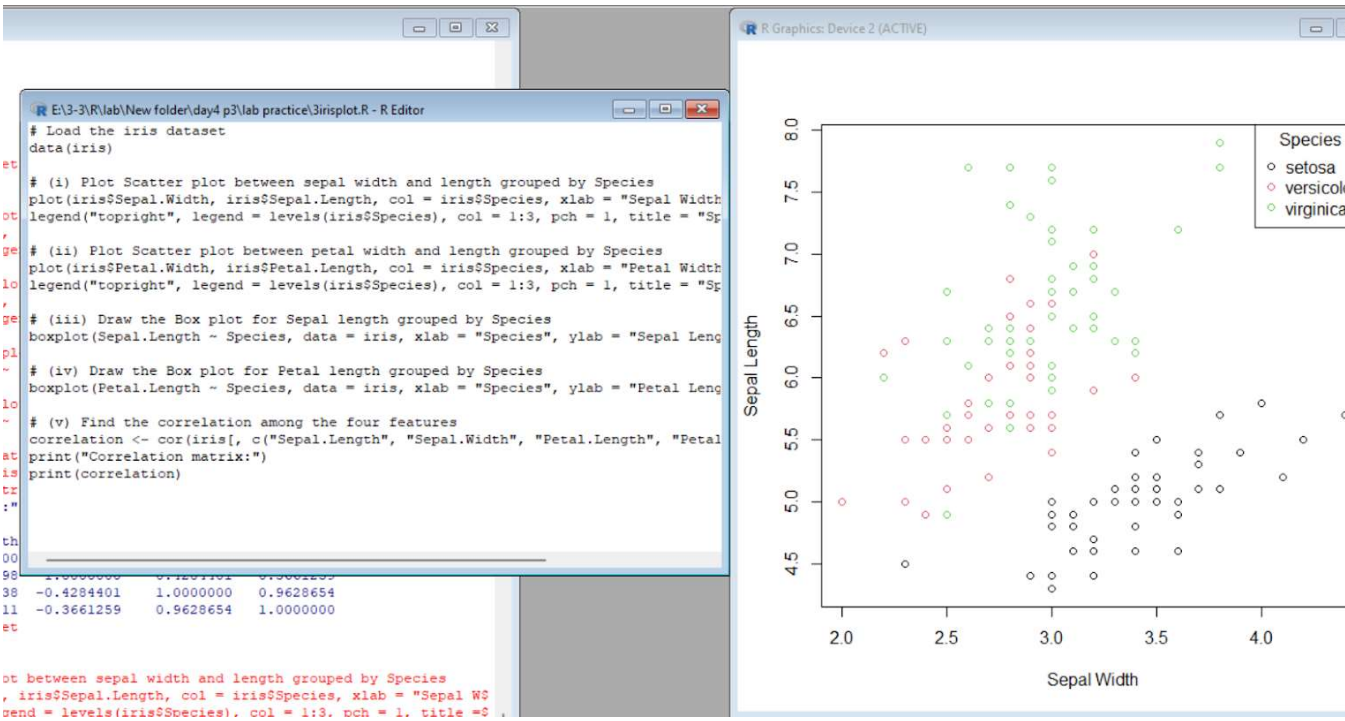
```
# Load the iris dataset
```

data(iris)

(i) Plot Scatter plot between sepal width and length grouped by Species

plot(iris$Sepal.Width, iris$Sepal.Length, col = iris$Species, xlab = "Sepal Width", ylab = "Sepal Length")

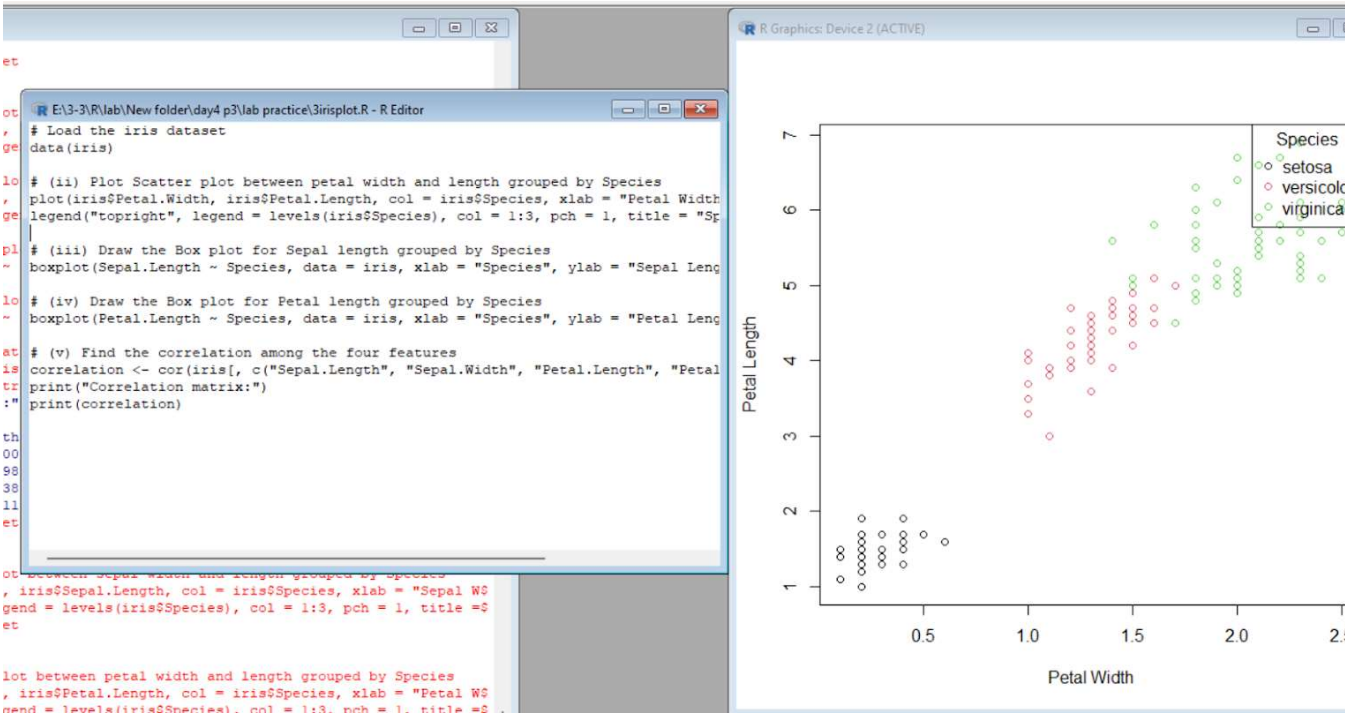
legend("topright", legend = levels(iris$Species), col = 1:3, pch = 1, title = "Species")



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

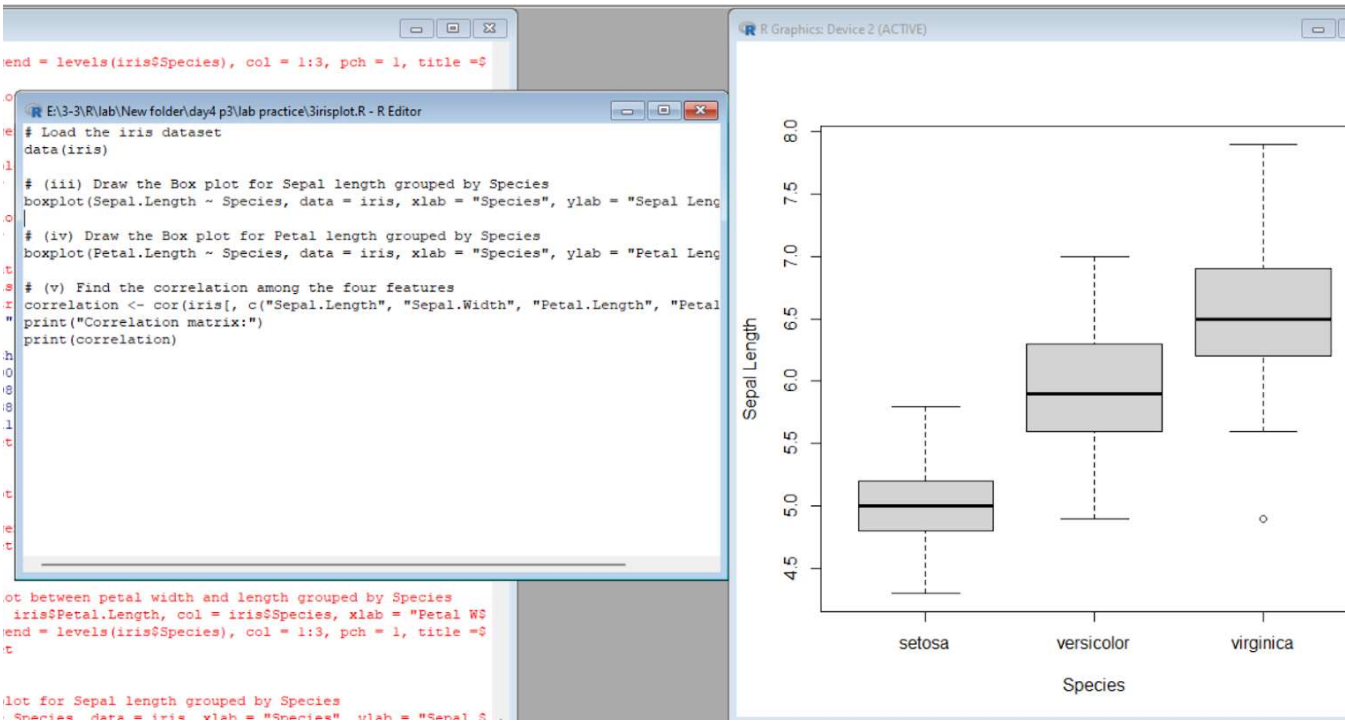
plot(iris$Petal.Width, iris$Petal.Length, col = iris$Species, xlab = "Petal Width", ylab = "Petal Length")

legend("topright", legend = levels(iris$Species), col = 1:3, pch = 1, title = "Species")



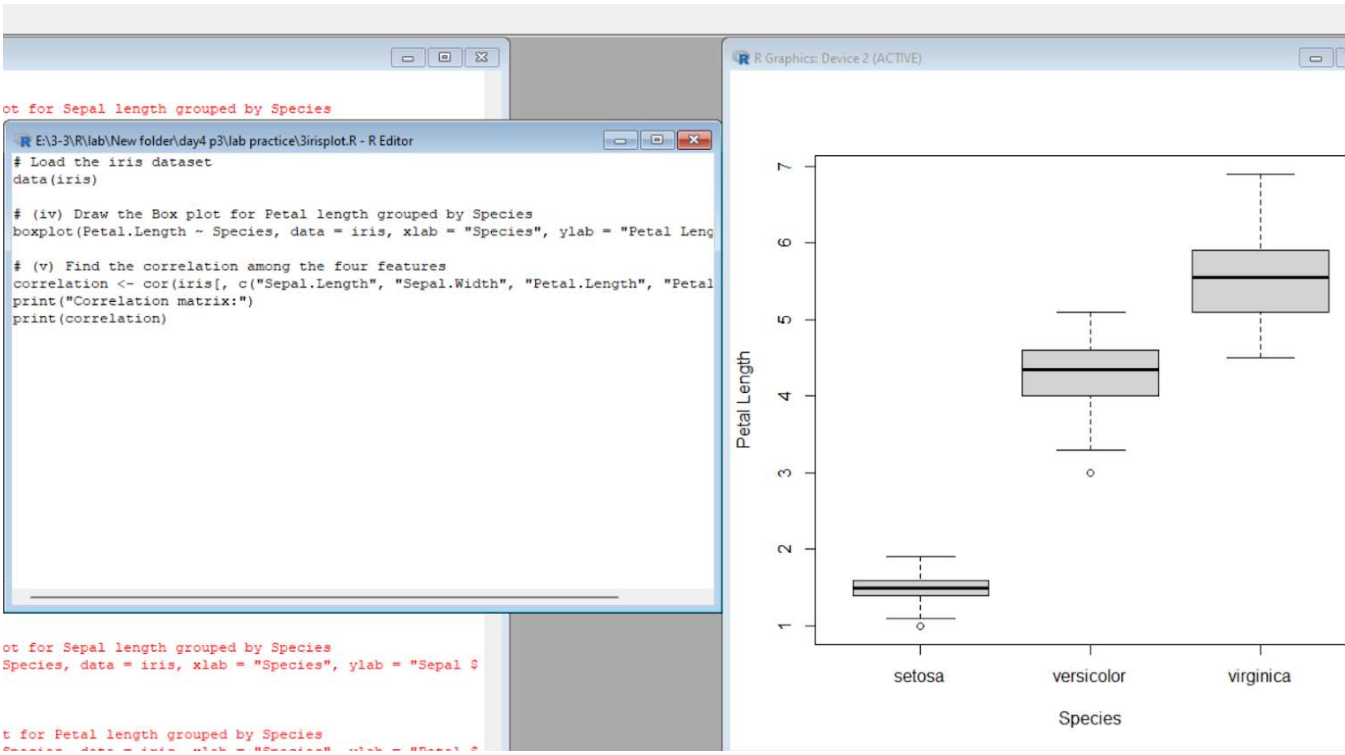
(iii) Draw the Box plot for Sepal length grouped by Species

boxplot(Sepal.Length ~ Species, data = iris, xlab = "Species", ylab = "Sepal Length")

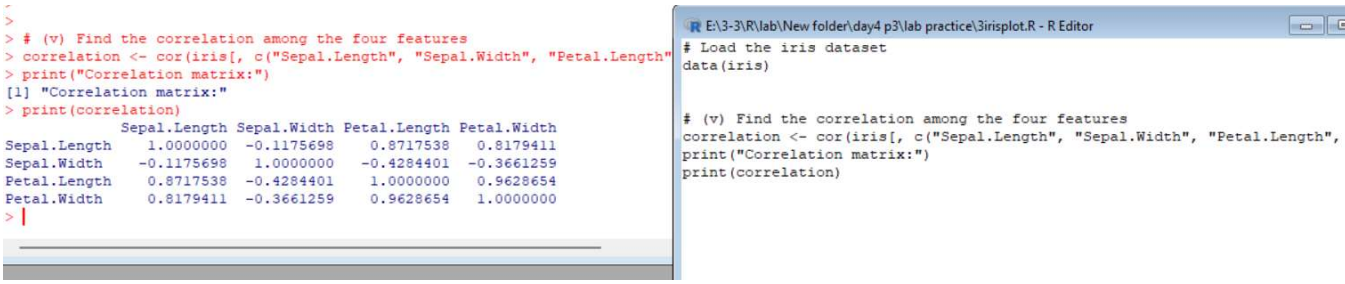


(iv) Draw the Box plot for Petal length grouped by Species

boxplot(Petal.Length ~ Species, data = iris, xlab = "Species", ylab = "Petal Length")



# (v) Find the correlation among the four features
correlation <- cor(iris[, c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")])
print("Correlation matrix:")
print(correlation)

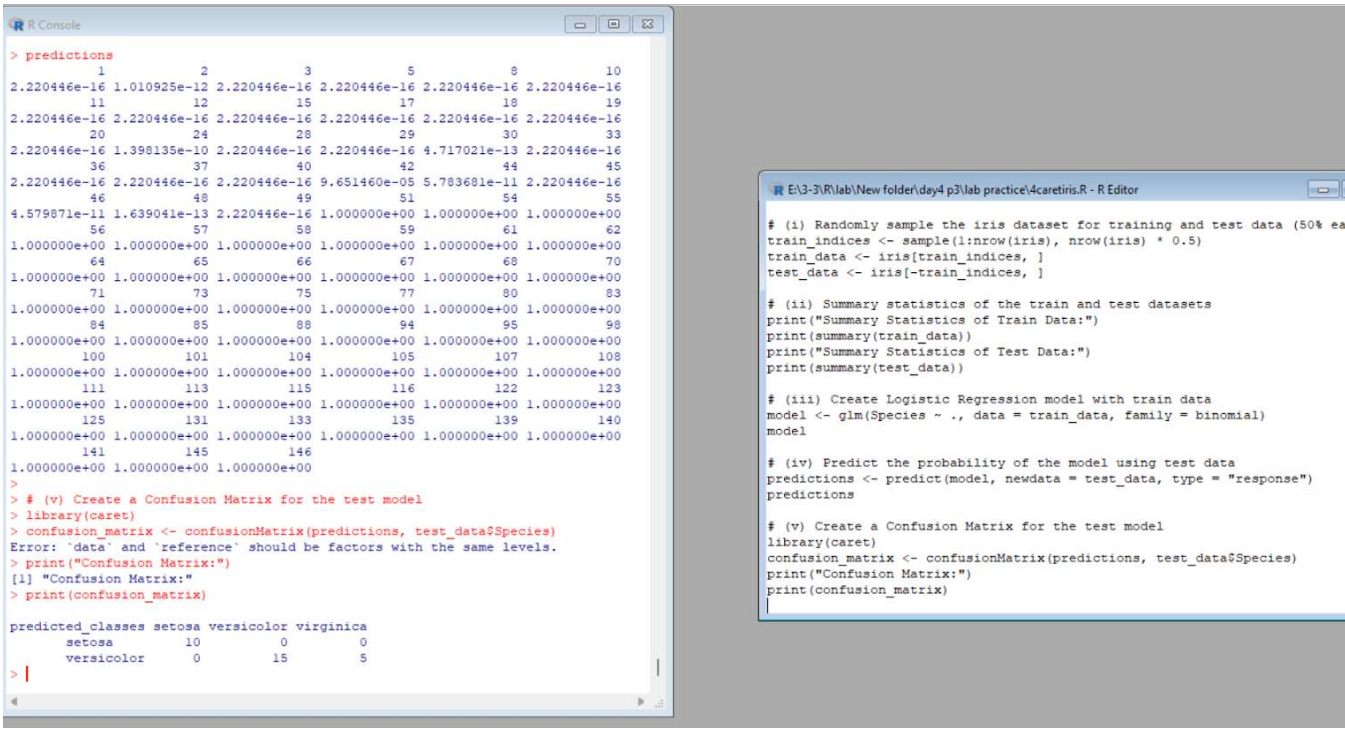


4.(i) Randomly Sample the iris dataset such as 50% data for training and 50% for test
(ii)find summary statistics of above train and test dataset.
(iii)Create Logistics regression with train data
(iv)Predict the probability of the model using test data
(v)Create Confusion matrix for above test model

SOURCE CODE:

Load the iris dataset

```
data(iris)

# Set seed for reproducibility
set.seed(123)

# (i) Randomly sample the iris dataset for training and test data (50% each)
train_indices <- sample(1:nrow(iris), nrow(iris) * 0.5)
train_data <- iris[train_indices, ]
test_data <- iris[-train_indices, ]

# (ii) Summary statistics of the train and test datasets
print("Summary Statistics of Train Data:")
print(summary(train_data))
print("Summary Statistics of Test Data:")
print(summary(test_data))

# (iii) Create Logistic Regression model with train data
model <- glm(Species ~ ., data = train_data, family = binomial)
model

# (iv) Predict the probability of the model using test data
predictions <- predict(model, newdata = test_data, type = "response")
predictions

# (v) Create a Confusion Matrix for the test model
library(caret)
confusion_matrix <- confusionMatrix(predictions, test_data$Species)
print("Confusion Matrix:")
print(confusion_matrix)
```



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.

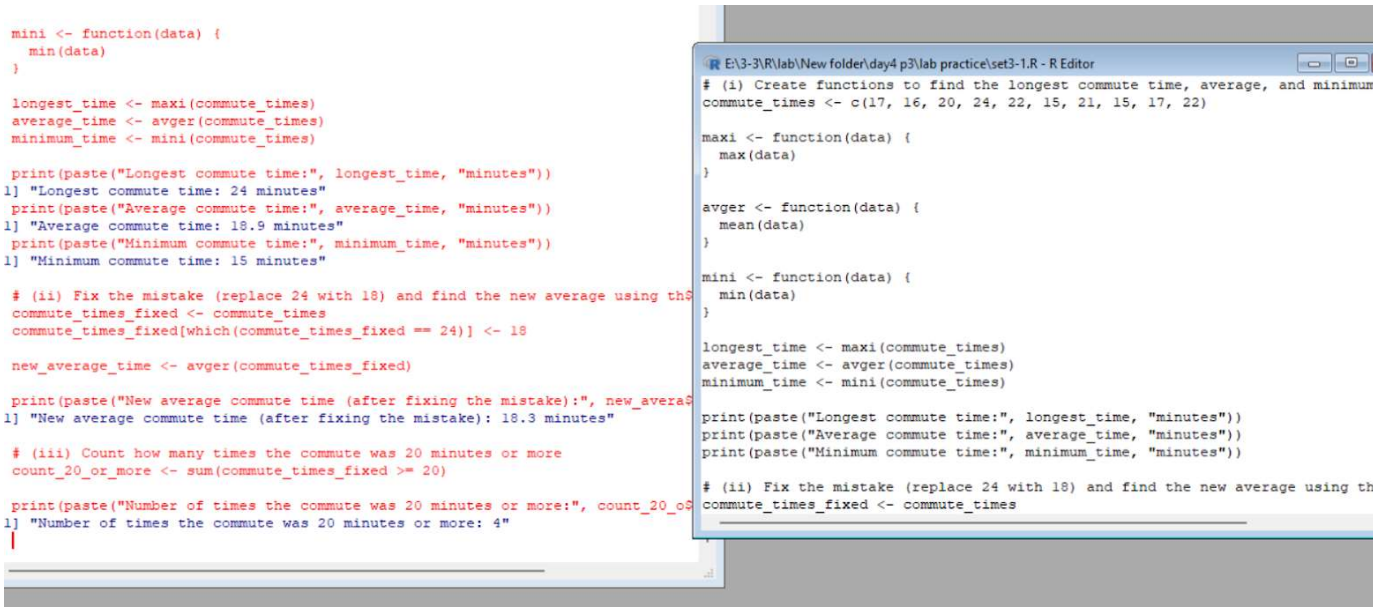
   (ii)Oops, the 24 was a mistake. It should have been 18. How can you fix this? Do so, and then find the new average using above functions.

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

   SOURCE CODE:

```
# (i) Create functions to find the longest commute time, average, and minimum
commute_times <- c(17, 16, 20, 24, 22, 15, 21, 15, 17, 22)

maxi <- function(data) {
```

```
  max(data)
}

avger <- function(data) {
  mean(data)
}

mini <- function(data) {
  min(data)
}

longest_time <- maxi(commute_times)
average_time <- avger(commute_times)
minimum_time <- mini(commute_times)

print(paste("Longest commute time:", longest_time, "minutes"))
print(paste("Average commute time:", average_time, "minutes"))
print(paste("Minimum commute time:", minimum_time, "minutes"))

# (ii) Fix the mistake (replace 24 with 18) and find the new average using the above functions
commute_times_fixed <- commute_times
commute_times_fixed[which(commute_times_fixed == 24)] <- 18

new_average_time <- avger(commute_times_fixed)

print(paste("New average commute time (after fixing the mistake):", new_average_time,
"minutes"))

# (iii) Count how many times the commute was 20 minutes or more
count_20_or_more <- sum(commute_times_fixed >= 20)

print(paste("Number of times the commute was 20 minutes or more:", count_20_or_more))
```



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).

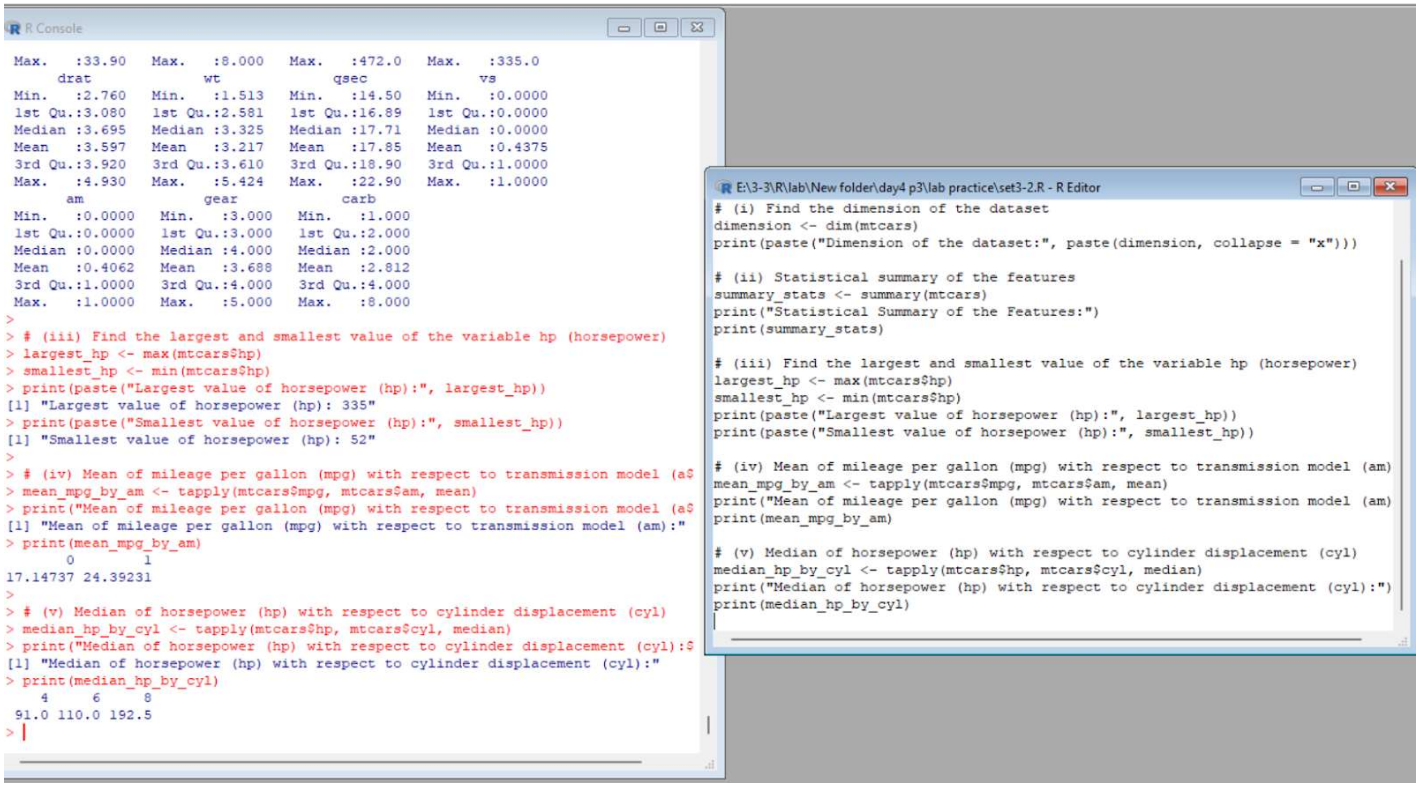
(iv)Give the mean of mileage per gallon (mpg) with respect to transmission model (feature named as 'am')

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

SOURCE CODE:

```
# Load the mtcars dataset

data(mtcars)



# (i) Find the dimension of the dataset

dimension <- dim(mtcars)

print(paste("Dimension of the dataset:", paste(dimension, collapse = "x")))



# (ii) Statistical summary of the features
```

```
summary_stats <- summary(mtcars)

print("Statistical Summary of the Features:")

print(summary_stats)


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

largest_hp <- max(mtcars$hp)

smallest_hp <- min(mtcars$hp)

print(paste("Largest value of horsepower (hp):", largest_hp))

print(paste("Smallest value of horsepower (hp):", smallest_hp))


# (iv) Mean of mileage per gallon (mpg) with respect to transmission model (am)

mean_mpg_by_am <- tapply(mtcars$mpg, mtcars$am, mean)

print("Mean of mileage per gallon (mpg) with respect to transmission model (am):")

print(mean_mpg_by_am)


# (v) Median of horsepower (hp) with respect to cylinder displacement (cyl)

median_hp_by_cyl <- tapply(mtcars$hp, mtcars$cyl, median)

print("Median of horsepower (hp) with respect to cylinder displacement (cyl):")

print(median_hp_by_cyl)
```



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')

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

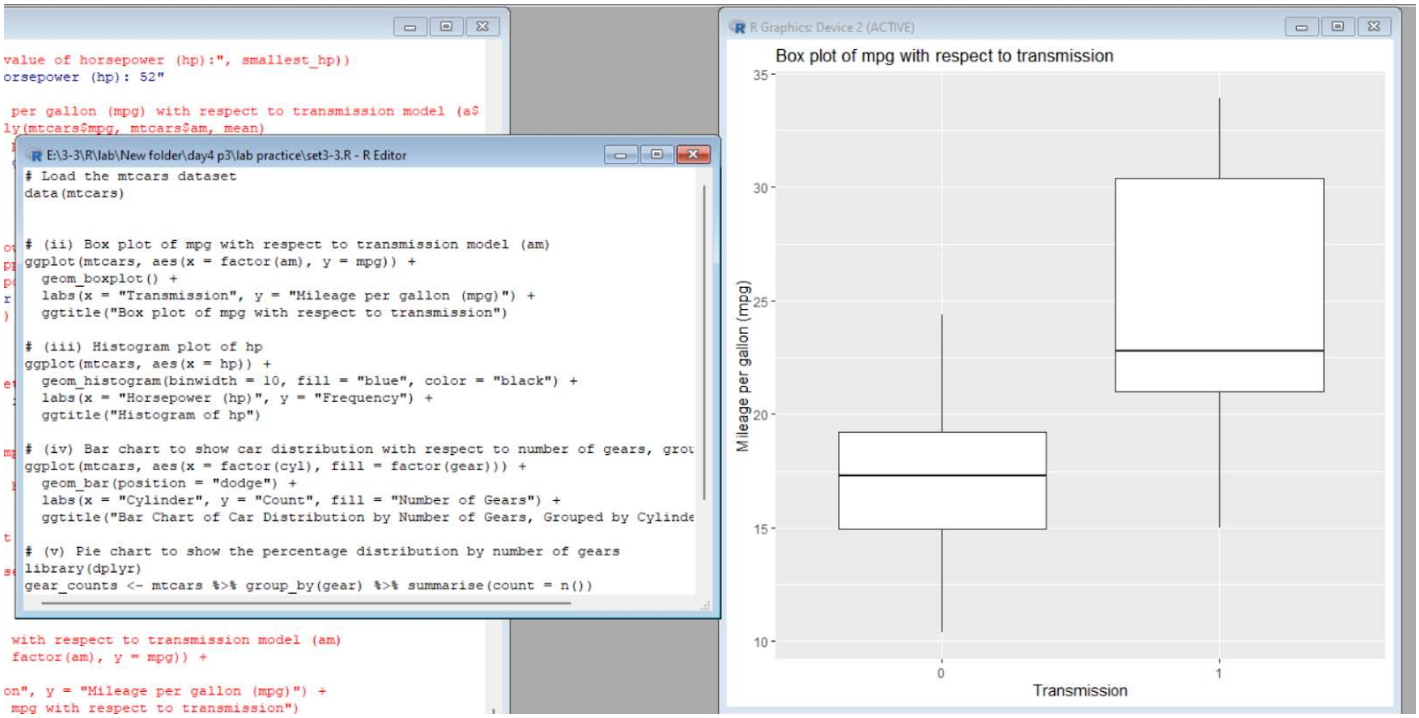
(iv)Draw the Bar Chart to show car distribution with respect to number of gears grouped by cylinder.(Grouped or multiple bar chart)

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

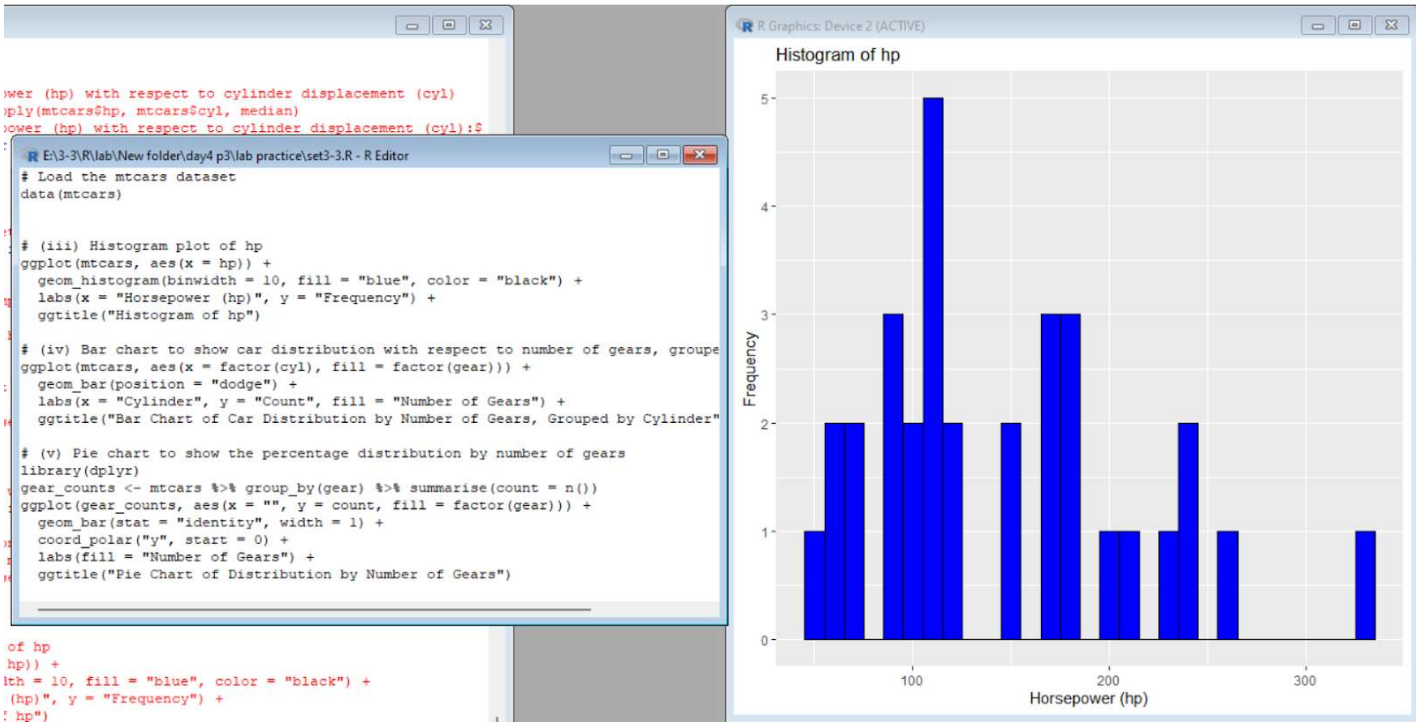
SOURCE CODE:

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

# (i) Scatter plot of mpg vs hp, grouped by transmission model (am)
library(ggplot2)
ggplot(mtcars, aes(x = hp, y = mpg, color = factor(am))) +
  geom_point() +
  labs(x = "Horsepower (hp)", y = "Mileage per gallon (mpg)", color = "Transmission") +
  ggtitle("Scatter plot of mpg vs hp, grouped by transmission")
```

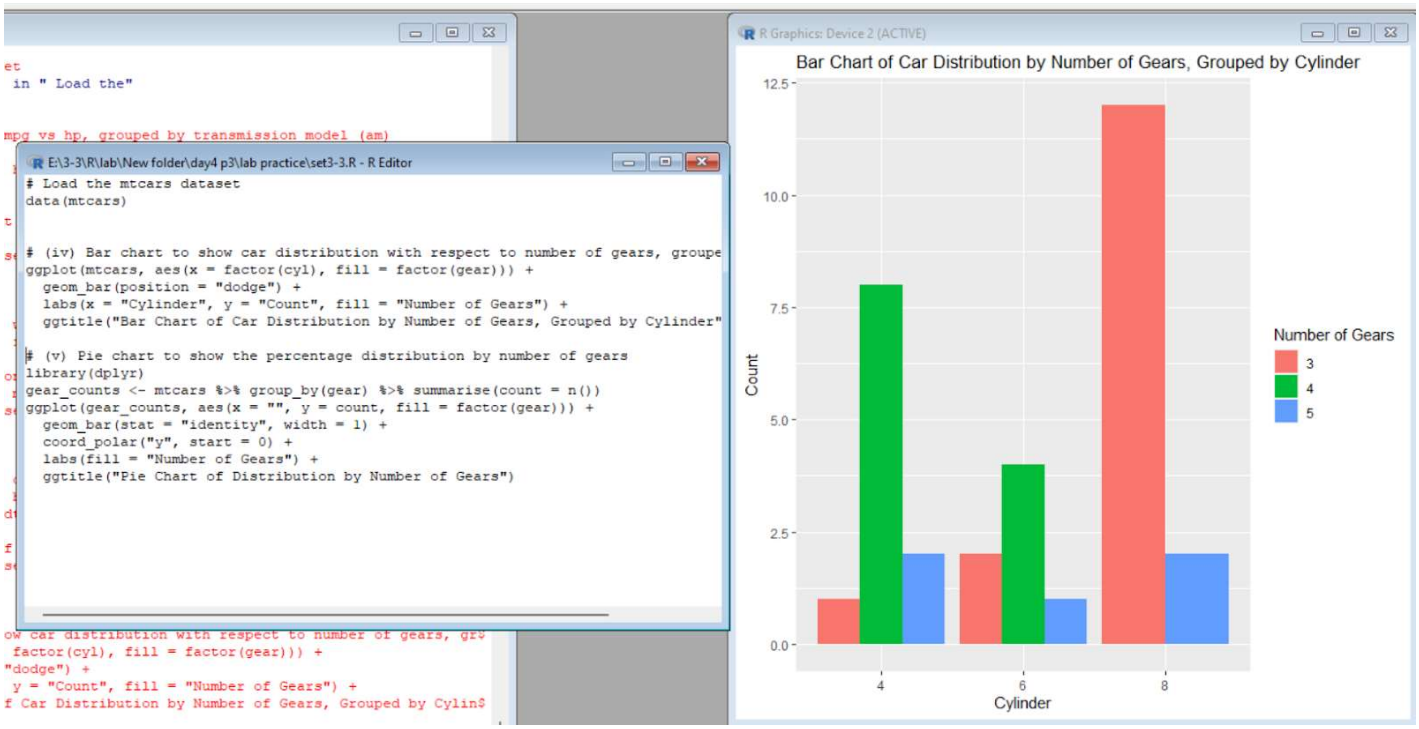

```
# (ii) Box plot of mpg with respect to transmission model (am)
ggplot(mtcars, aes(x = factor(am), y = mpg)) +
  geom_boxplot() +
  labs(x = "Transmission", y = "Mileage per gallon (mpg)") +
  ggtitle("Box plot of mpg with respect to transmission")
```



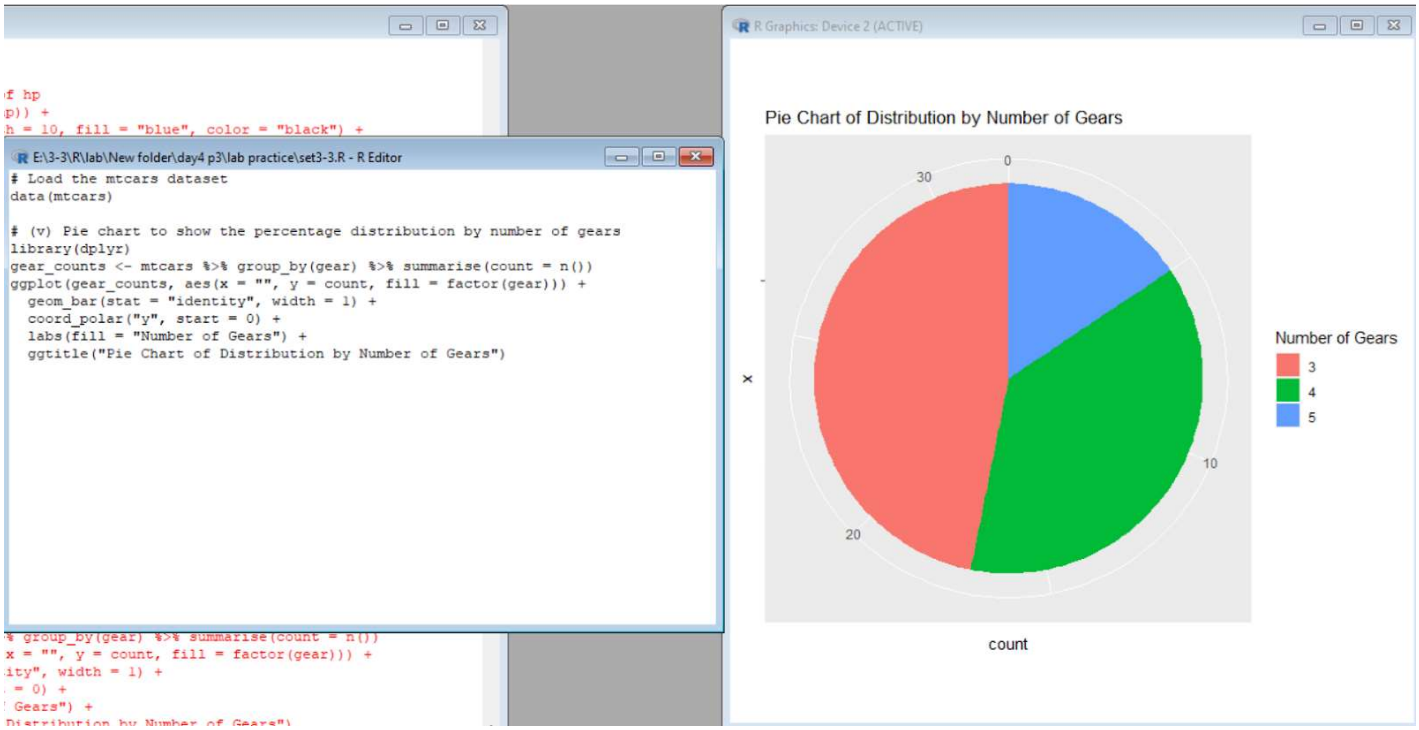

```
# (iii) Histogram plot of hp
ggplot(mtcars, aes(x = hp)) +
  geom_histogram(binwidth = 10, fill = "blue", color = "black") +
  labs(x = "Horsepower (hp)", y = "Frequency") +
  ggtitle("Histogram of hp")
```




```
# (iv) Bar chart to show car distribution with respect to number of gears, grouped by cylinder
ggplot(mtcars, aes(x = factor(cyl), fill = factor(gear))) +
  geom_bar(position = "dodge") +
  labs(x = "Cylinder", y = "Count", fill = "Number of Gears") +
  ggtitle("Bar Chart of Car Distribution by Number of Gears, Grouped by Cylinder")
```




```
# (v) Pie chart to show the percentage distribution by number of gears
library(dplyr)
gear_counts <- mtcars %>% group_by(gear) %>% summarise(count = n())
ggplot(gear_counts, aes(x = "", y = count, fill = factor(gear))) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y", start = 0) +
  labs(fill = "Number of Gears") +
  ggtitle("Pie Chart of Distribution by Number of Gears")
```



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.

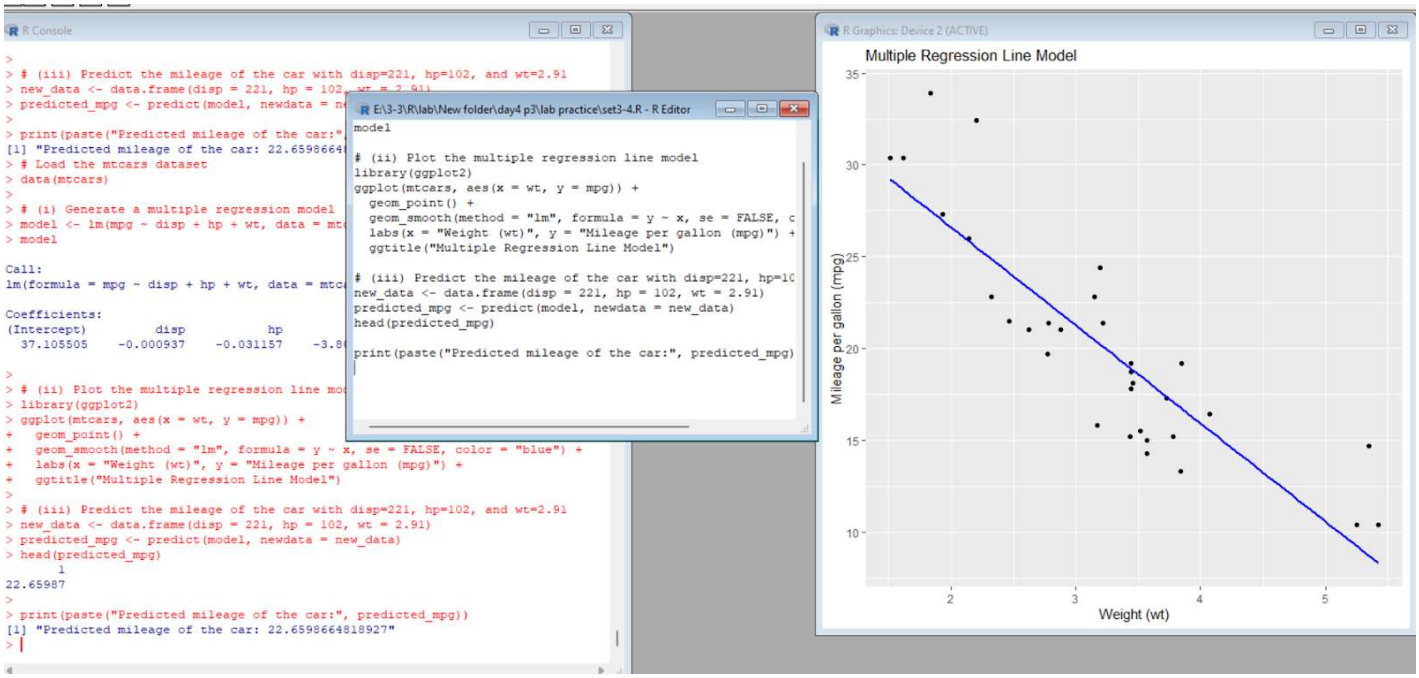
(iii) Predict the mileage of the car with dsp=221, hp=102 and wt=2.91

SOURCE CODE:

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

# (i) Generate a multiple regression model
model <- lm(mpg ~ disp + hp + wt, data = mtcars)

# (ii) Plot the multiple regression line model
library(ggplot2)
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE, color = "blue") +
  labs(x = "Weight (wt)", y = "Mileage per gallon (mpg)") +
  ggtitle("Multiple Regression Line Model")
```

# (iii) Predict the mileage of the car with disp=221, hp=102, and wt=2.91
new_data <- data.frame(disp = 221, hp = 102, wt = 2.91)
predicted_mpg <- predict(model, newdata = new_data)

print(paste("Predicted mileage of the car:", predicted_mpg))



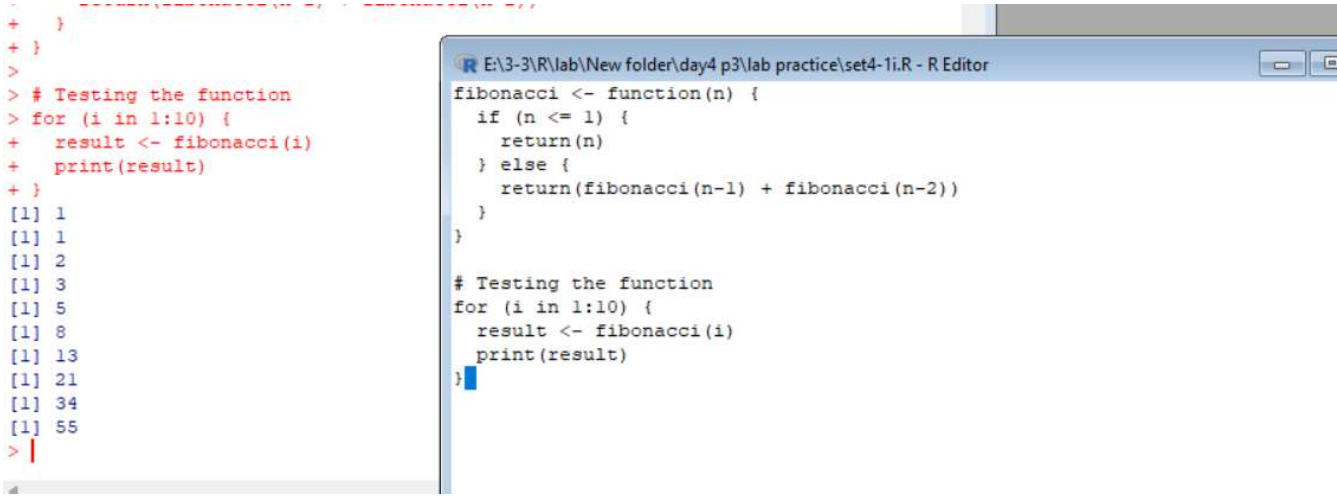
Set IV

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

   SOURCE CODE:

```
fibonacci <- function(n) {
  if (n <= 1) {
    return(n)
  } else {
    return(fibonacci(n-1) + fibonacci(n-2))
```

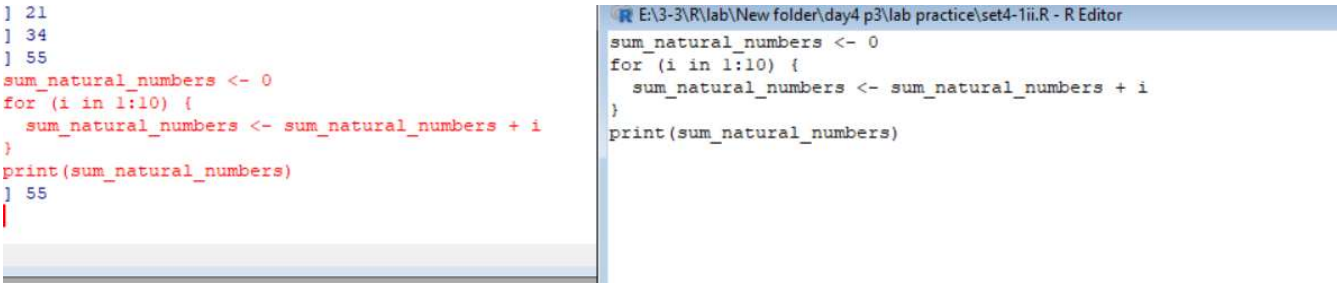
```
  }
}

# Testing the function
for (i in 1:10) {
  result <- fibonacci(i)
  print(result)
}
```



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

SOURCE CODE:

```
sum_natural_numbers <- 0
for (i in 1:10) {
  sum_natural_numbers <- sum_natural_numbers + i
}
print(sum_natural_numbers)
```



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

SOURCE CODE:

```
vector <- 1:10
squares <- list()
for (num in vector) {
  square <- num^2
```

```
  squares <- c(squares, square)
}
print(squares)
```

```
> vector <- 1:10
> squares <- list()
> for (num in vector) {
+   square <- num^2
+   squares <- c(squares, square)
+ }
> print(squares)
[[1]]
[1] 1

[[2]]
[1] 4

[[3]]
[1] 9

[[4]]
[1] 16

[[5]]
[1] 25

[[6]]
[1] 36

[[7]]
[1] 49

[[8]]
[1] 64

[[9]]
[1] 81

[[10]]
[1] 100

>
```

E:\3-3\R\lab\New folder\day4 p3\lab practice\SET4-1iii.R - R Editor

```
vector <- 1:10
squares <- list()
for (num in vector) {
  square <- num^2
  squares <- c(squares, square)
}
print(squares)
```

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.

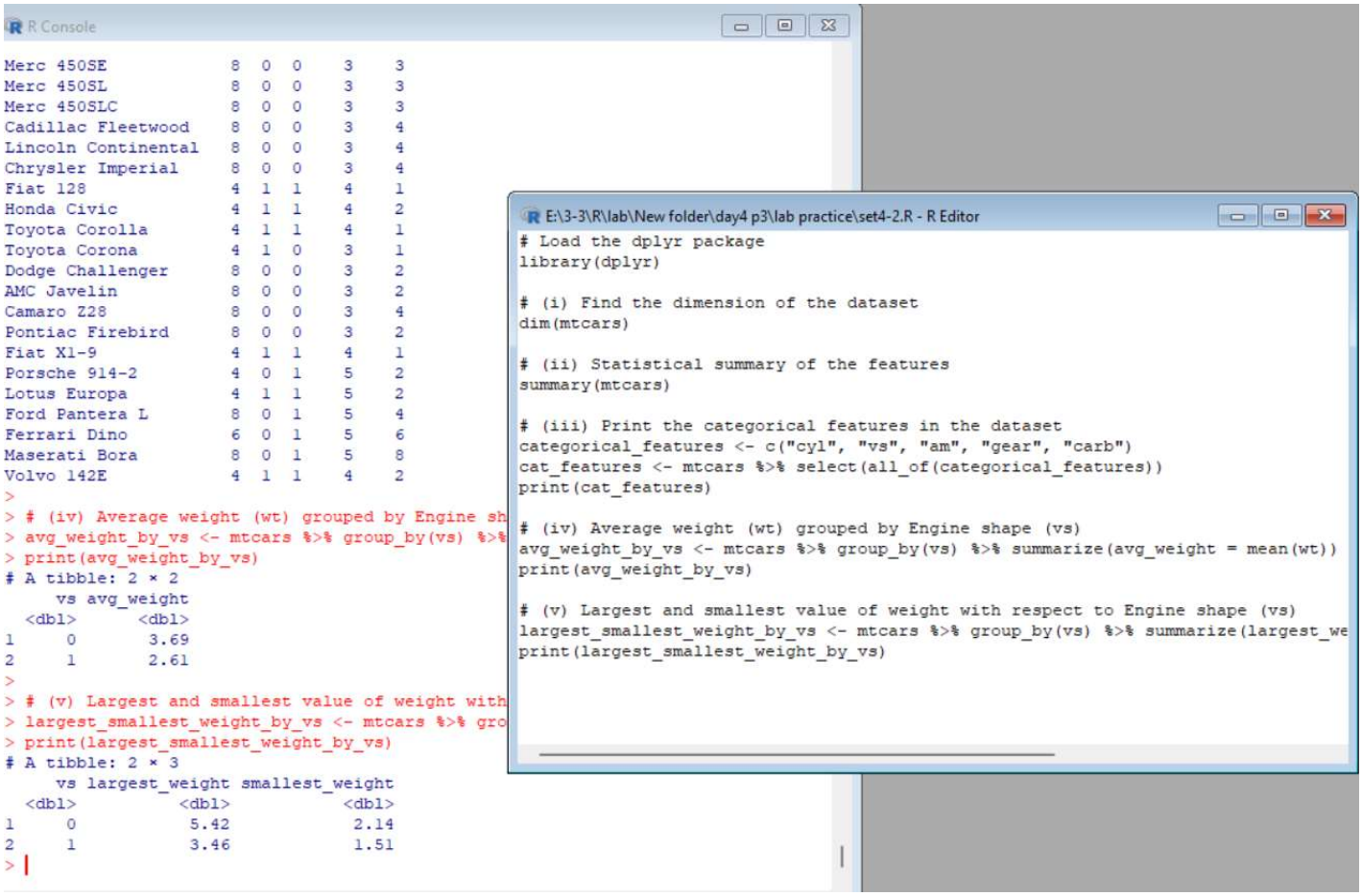
   (iii)Print the categorical features in Dataset

   (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

SOURCE CODE:

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

# (i) Find the dimension of the dataset
dim(mtcars)

# (ii) Statistical summary of the features
summary(mtcars)

# (iii) Print the categorical features in the dataset
categorical_features <- c("cyl", "vs", "am", "gear", "carb")
cat_features <- mtcars %>% select(all_of(categorical_features))
print(cat_features)

# (iv) Average weight (wt) grouped by Engine shape (vs)
avg_weight_by_vs <- mtcars %>% group_by(vs) %>% summarize(avg_weight = mean(wt))
print(avg_weight_by_vs)

# (v) Largest and smallest value of weight with respect to Engine shape (vs)
largest_smallest_weight_by_vs <- mtcars %>% group_by(vs) %>% summarize(largest_weight = max(wt), smallest_weight = min(wt))
print(largest_smallest_weight_by_vs)
```



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)

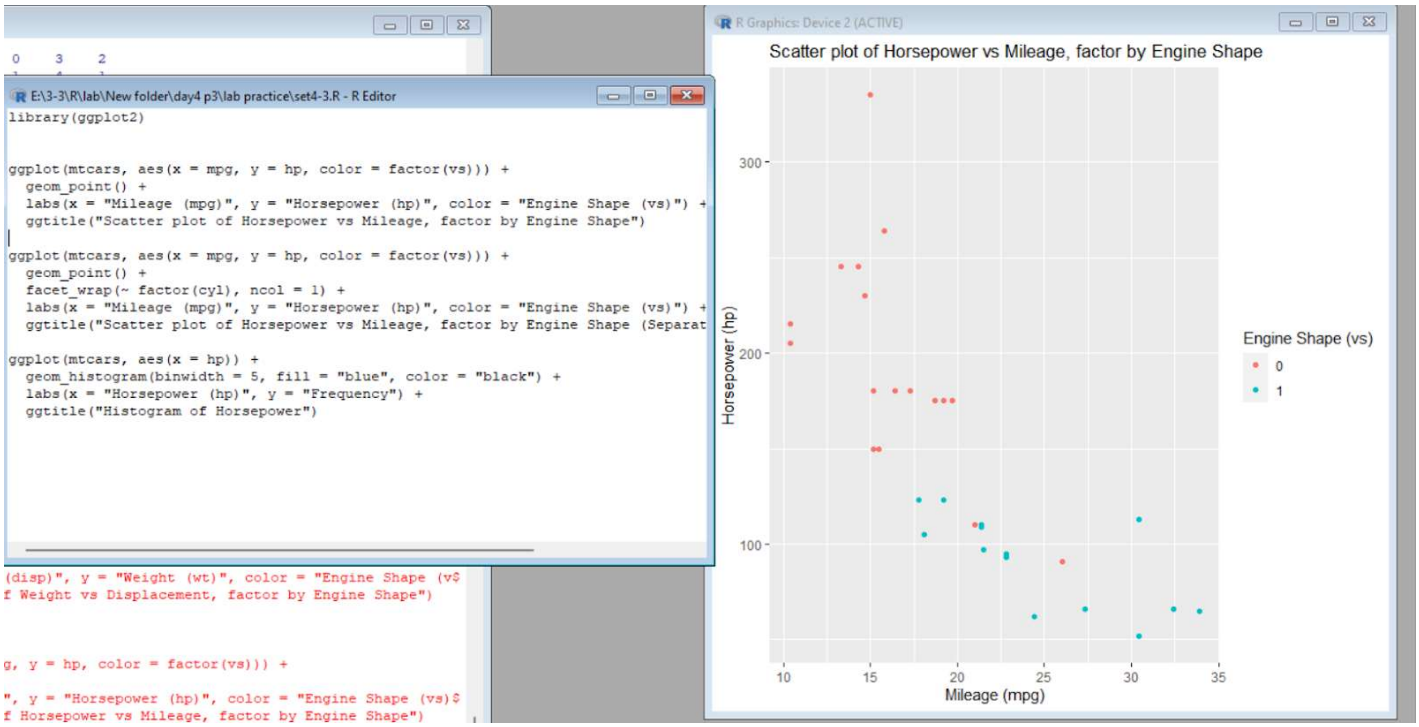
(iv)In above(ii) plot , Separate columns according to cylinders(cyl) size

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

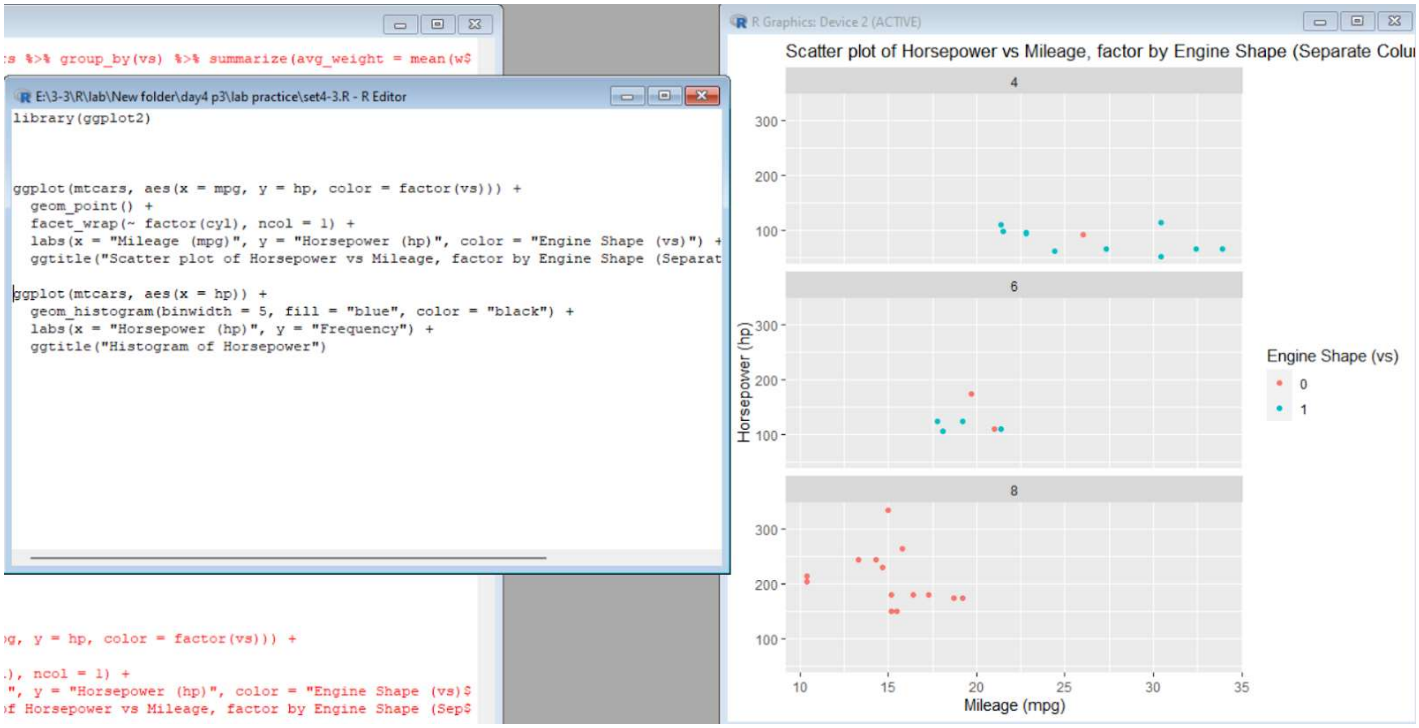
SOURCE CODE:

```
library(ggplot2)

ggplot(mtcars, aes(x = disp, y = wt, color = factor(vs))) +

  geom_point() +

  labs(x = "Displacement (disp)", y = "Weight (wt)", color = "Engine Shape (vs)") +

  ggtitle("Scatter plot of Weight vs Displacement, factor by Engine Shape")
```

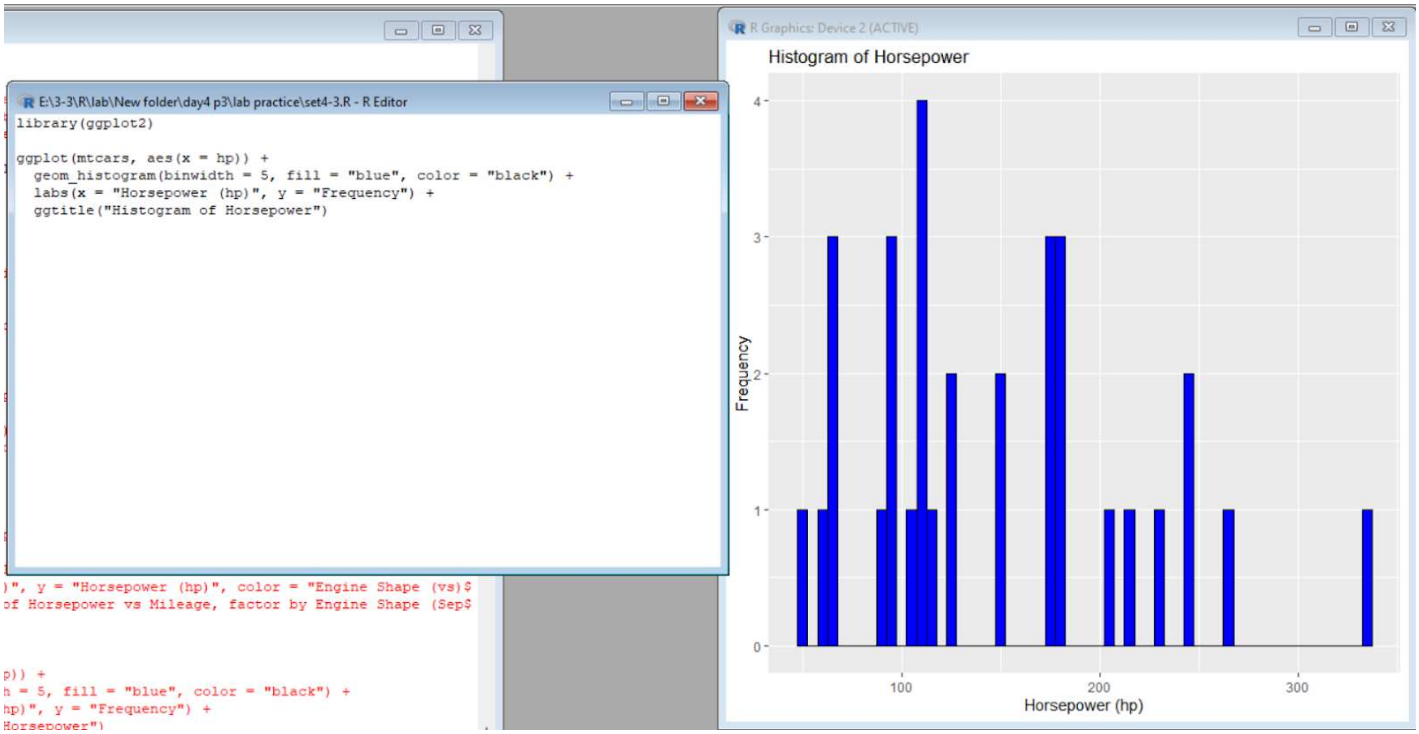

```
ggplot(mtcars, aes(x = mpg, y = hp, color = factor(vs))) +
  geom_point() +
  labs(x = "Mileage (mpg)", y = "Horsepower (hp)", color = "Engine Shape (vs)") +
  ggtitle("Scatter plot of Horsepower vs Mileage, factor by Engine Shape")
```




```
ggplot(mtcars, aes(x = mpg, y = hp, color = factor(vs))) +
  geom_point() +
  facet_wrap(~ factor(cyl), ncol = 1) +
  labs(x = "Mileage (mpg)", y = "Horsepower (hp)", color = "Engine Shape (vs)") +
  ggtitle("Scatter plot of Horsepower vs Mileage, factor by Engine Shape (Separate Columns by Cylinder)")
```




```
ggplot(mtcars, aes(x = hp)) +
  geom_histogram(binwidth = 5, fill = "blue", color = "black") +
  labs(x = "Horsepower (hp)", y = "Frequency") +
  ggtitle("Histogram of Horsepower")
```



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.

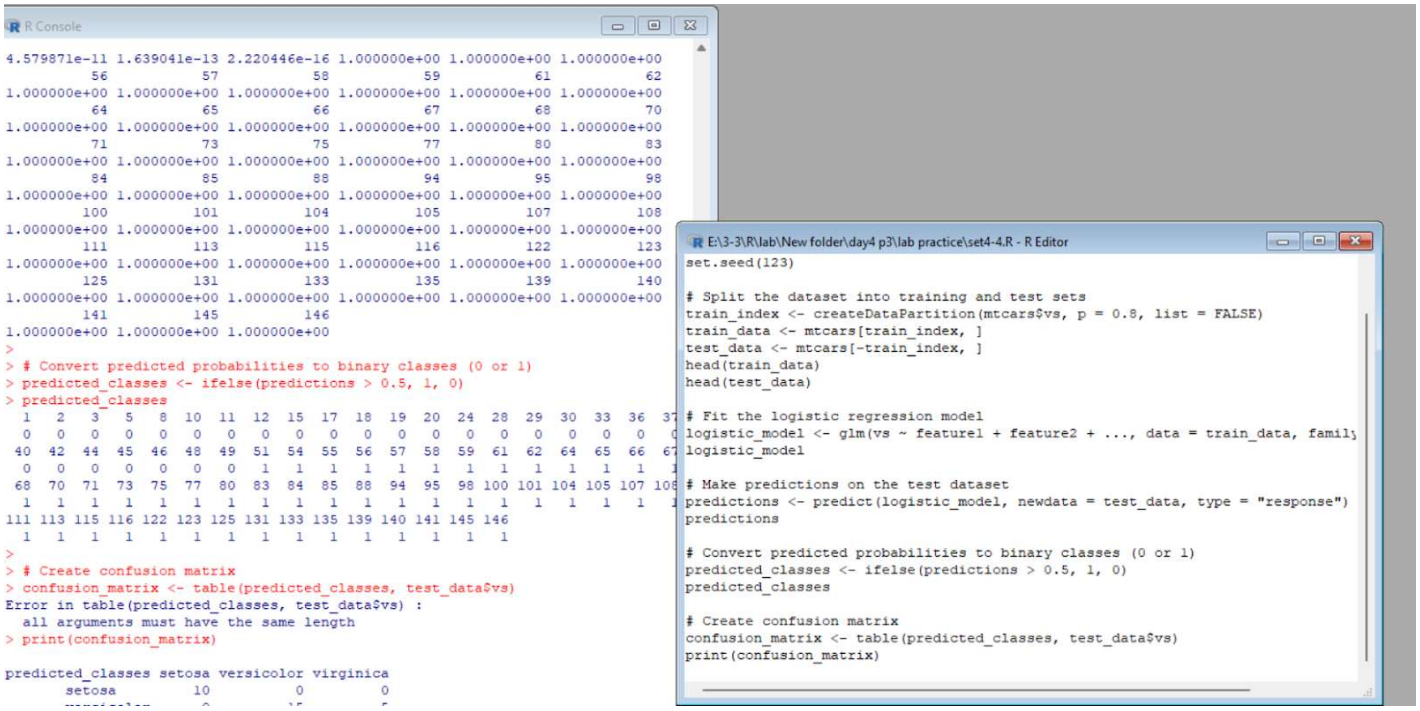
(ii) Split the data set randomly with 80:20 ration to create train and test dataset and create logistic model

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

SOURCE CODE:

```
library(caret)


# Set the seed for reproducibility

set.seed(123)


# Split the dataset into training and test sets

train_index <- createDataPartition(mtcars$vs, p = 0.8, list = FALSE)
```

```
train_data <- mtcars[train_index, ]
test_data <- mtcars[-train_index, ]
head(train_data)
head(test_data)

# Fit the logistic regression model
logistic_model <- glm(vs ~ feature1 + feature2 + ..., data = train_data, family = binomial)
logistic_model

# Make predictions on the test dataset
predictions <- predict(logistic_model, newdata = test_data, type = "response")
predictions

# Convert predicted probabilities to binary classes (0 or 1)
predicted_classes <- ifelse(predictions > 0.5, 1, 0)
predicted_classes

# Create confusion matrix
confusion_matrix <- table(predicted_classes, 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.)

(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

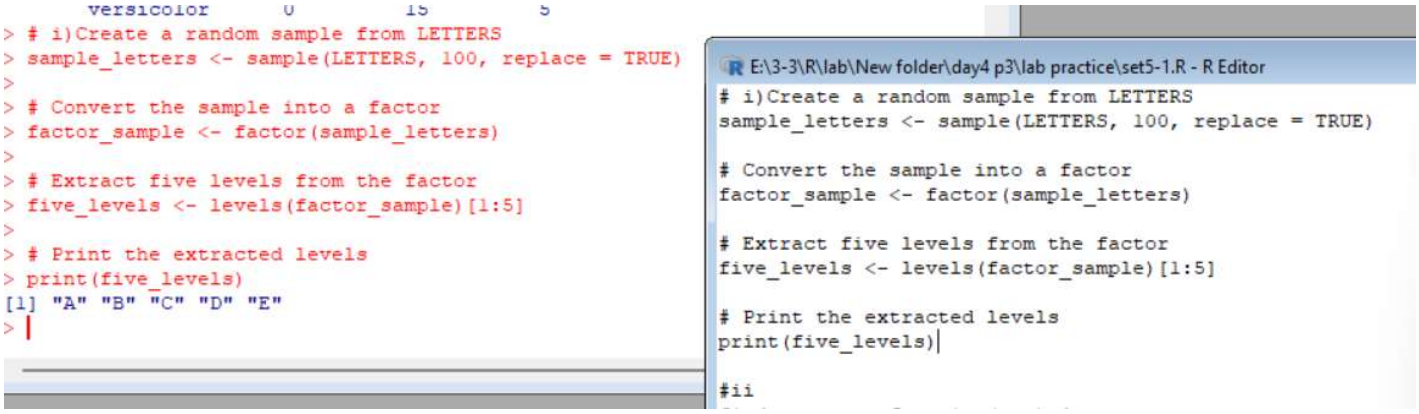
(iii)Wirte the R function to find the number of vowels in given string

Sample input c<- “matrix”, output<-2

SOURCE CODE:

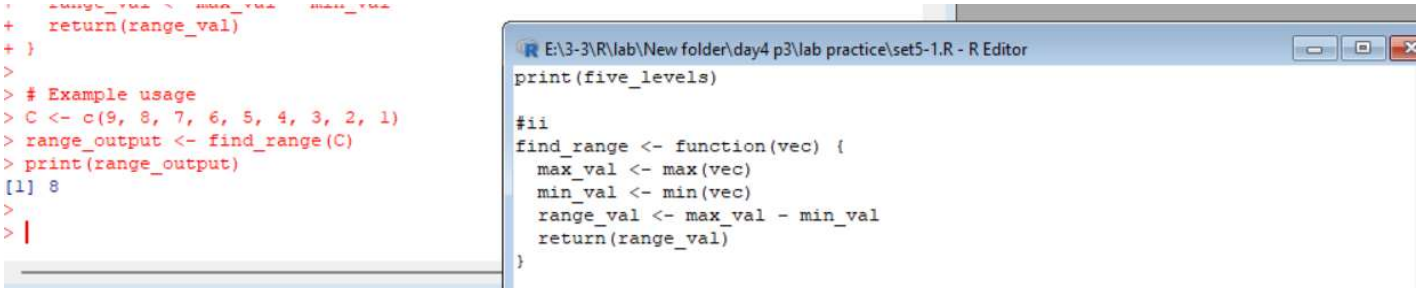
i)Create a random sample from LETTERS

```
sample_letters <- sample(LETTERS, 100, replace = TRUE)


# Convert the sample into a factor

factor_sample <- factor(sample_letters

# Extract five levels from the factor

five_levels <- levels(factor_sample)[1:5]

# Print the extracted levels

print(five_levels)
```

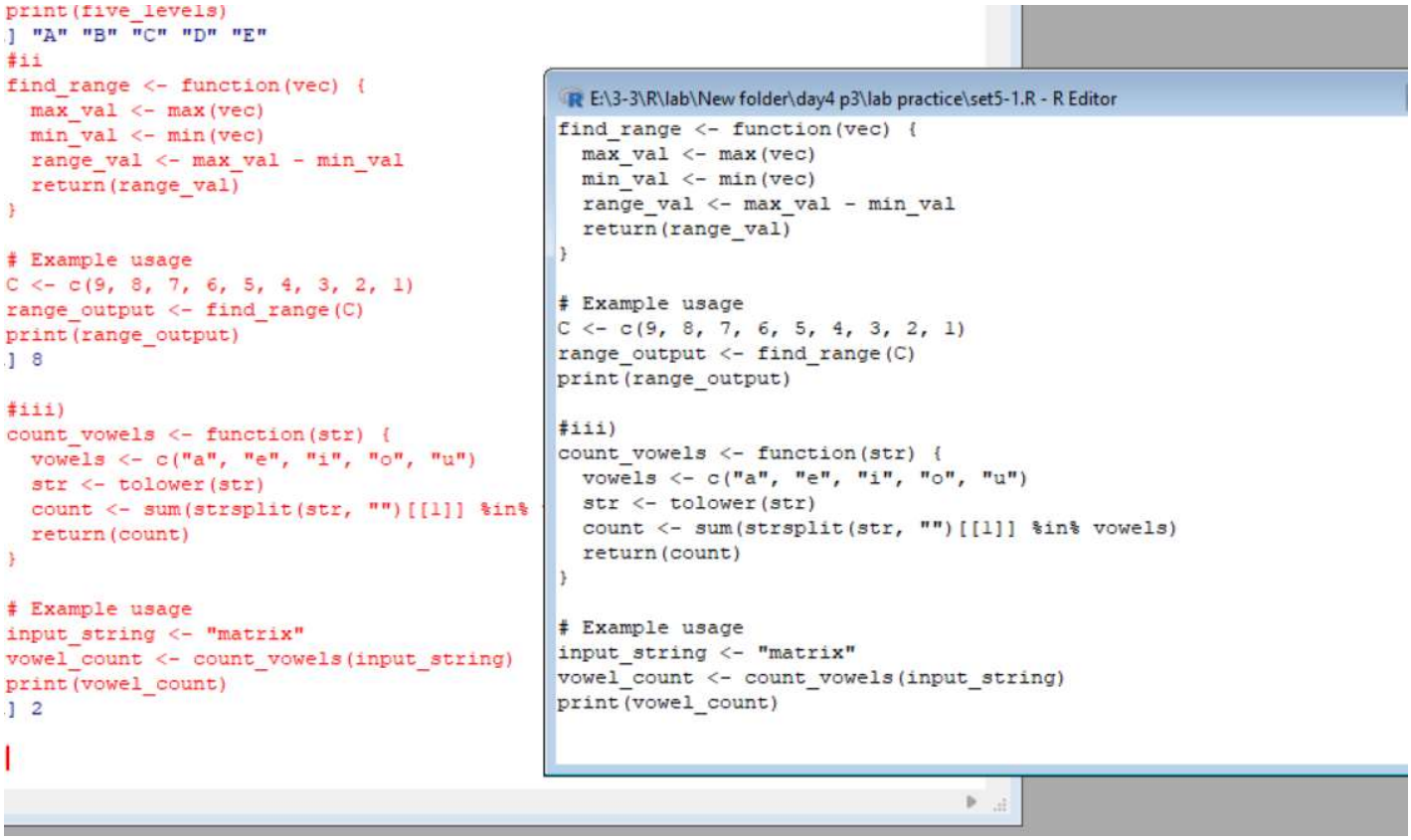



```
#ii

find_range <- function(vec) {

  max_val <- max(vec)

  min_val <- min(vec)

  range_val <- max_val - min_val

  return(range_val)

}

# Example usage
```

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

range_output <- find_range(C)

print(range_output)
```




```
#iii)

count_vowels <- function(str) {

  vowels <- c("a", "e", "i", "o", "u")

  str <- tolower(str)

  count <- sum(strsplit(str, "")[[1]] %in% vowels)

  return(count)

}

# Example usage

input_string <- "matrix"

vowel_count <- count_vowels(input_string)

print(vowel_count)
```



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

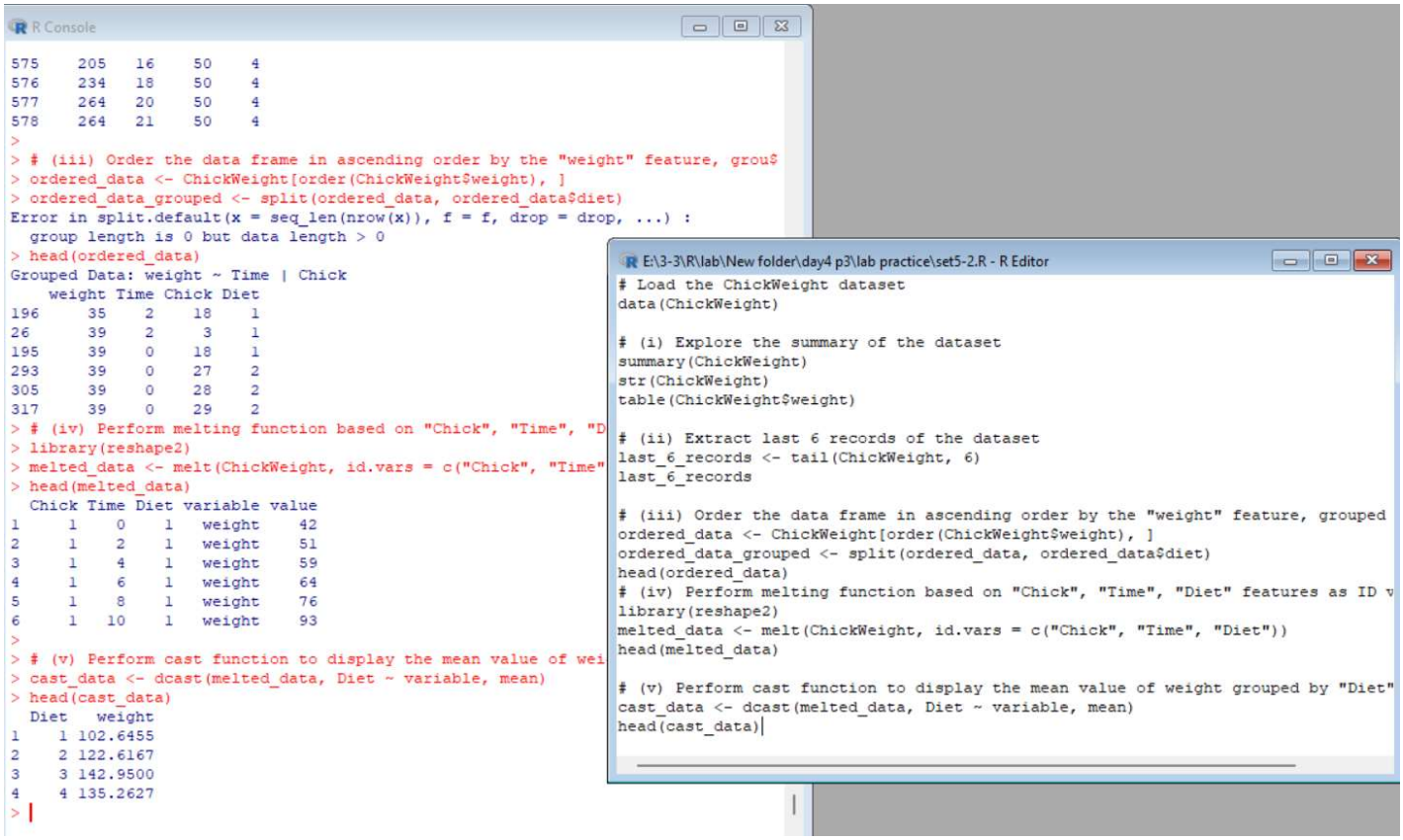
(v)Perform cast function to display the mean value of weight grouped by Diet

SOURCE CODE:

Load the ChickWeight dataset

data(ChickWeight)

```
# (i) Explore the summary of the dataset
summary(ChickWeight)
str(ChickWeight)
table(ChickWeight$weight)

# (ii) Extract last 6 records of the dataset
last_6_records <- tail(ChickWeight, 6)

# (iii) Order the data frame in ascending order by the "weight" feature, grouped by "diet"
ordered_data <- ChickWeight[order(ChickWeight$weight), ]
ordered_data_grouped <- split(ordered_data, ordered_data$diet)

# (iv) Perform melting function based on "Chick", "Time", "Diet" features as ID variables
library(reshape2)
melted_data <- melt(ChickWeight, id.vars = c("Chick", "Time", "Diet"))

# (v) Perform cast function to display the mean value of weight grouped by "Diet"
cast_data <- dcast(melted_data, Diet ~ variable, mean)
```



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

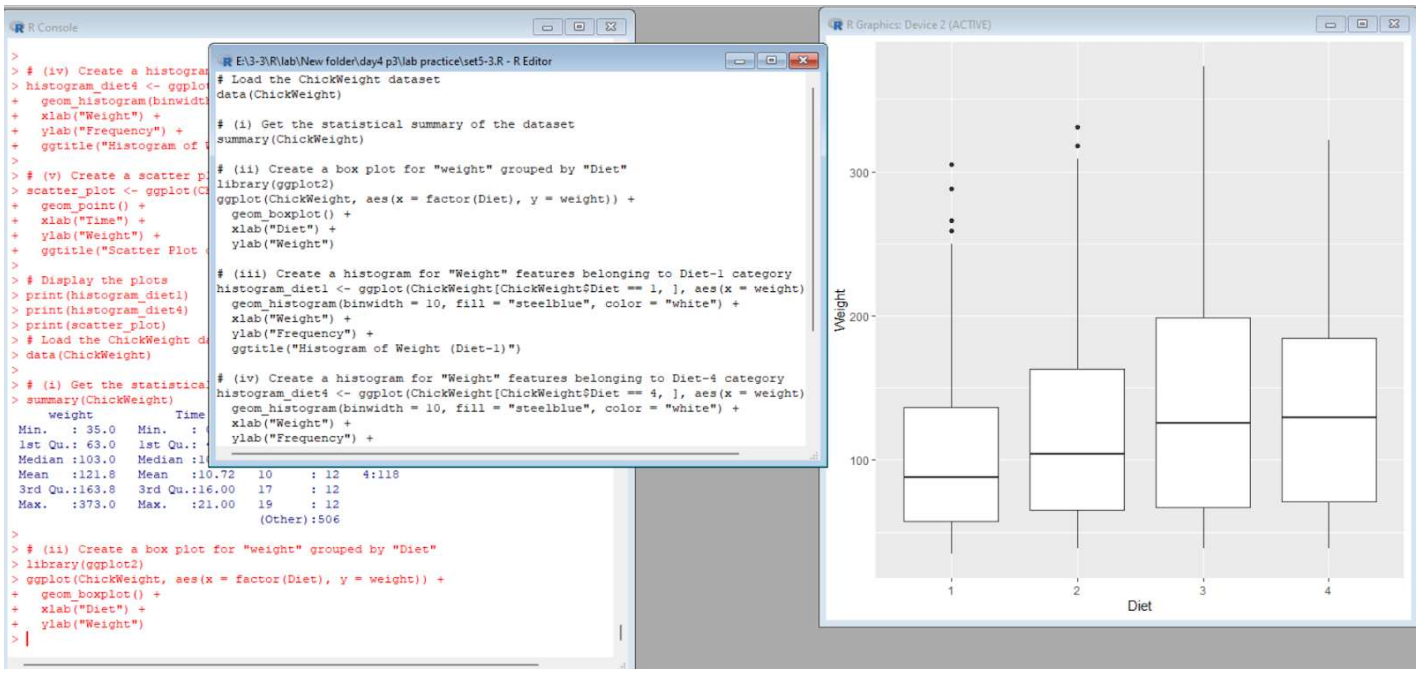
(v) Create Scatter plot for weight vs Time grouped by Diet

SOURCE CODE:

```
# Load the ChickWeight dataset

data(ChickWeight)


# (i) Get the statistical summary of the dataset

summary(ChickWeight)
```

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

```
library(ggplot2)

ggplot(ChickWeight, aes(x = factor(Diet), y = weight)) +

  geom_boxplot() +

  xlab("Diet") +

  ylab("Weight")
```



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

```
histogram_diet1 <- ggplot(ChickWeight[ChickWeight$Diet == 1, ], aes(x = weight)) +

  geom_histogram(binwidth = 10, fill = "steelblue", color = "white") +

  xlab("Weight") +

  ylab("Frequency") +

  ggtitle("Histogram of Weight (Diet-1)")
```


```
# (iv) Create a histogram for "Weight" features belonging to Diet-4 category
histogram_diet4 <- ggplot(ChickWeight[ChickWeight$Diet == 4, ], aes(x = weight)) +
  geom_histogram(binwidth = 10, fill = "steelblue", color = "white") +
  xlab("Weight") +
  ylab("Frequency") +
  ggtitle("Histogram of Weight (Diet-4)")
```

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

scatter_plot <- ggplot(ChickWeight, aes(x = Time, y = weight, color = factor(Diet))) +

  geom_point() +

  xlab("Time") +

  ylab("Weight") +

  ggtitle("Scatter Plot of Weight vs Time (Grouped by Diet)")

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

SOURCE CODE:

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

# (i) Create a multiple regression model to predict weight using "Time" and "Diet" as predictor variables
model <- lm(weight ~ Time + Diet, data = ChickWeight)
model
# (ii) Predict weight for Time = 10 and Diet = 1
new_data <- data.frame(Time = 10, Diet = factor(1, levels = levels(ChickWeight$Diet)))
predicted_weight <- predict(model, newdata = new_data)
head(predicted_weight)
# (iii) Calculate the mean absolute error (MAE) in the model
actual_weight <- ChickWeight$weight[ChickWeight$Time == 10 & ChickWeight$Diet == 1]
mae <- mean(abs(actual_weight - predicted_weight))
mae
```

# Print the predicted weight and MAE
print(paste("Predicted weight:", predicted_weight))
print(paste("MAE:", mae))