SAVEETHA SCHOOL OF ENGINEERING,

SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES

ITA 0451 - STATISTICS WITH R PROGRAMMING

DAY 4 – LAB ASSESSMENT Part 3

Reg No:192011283

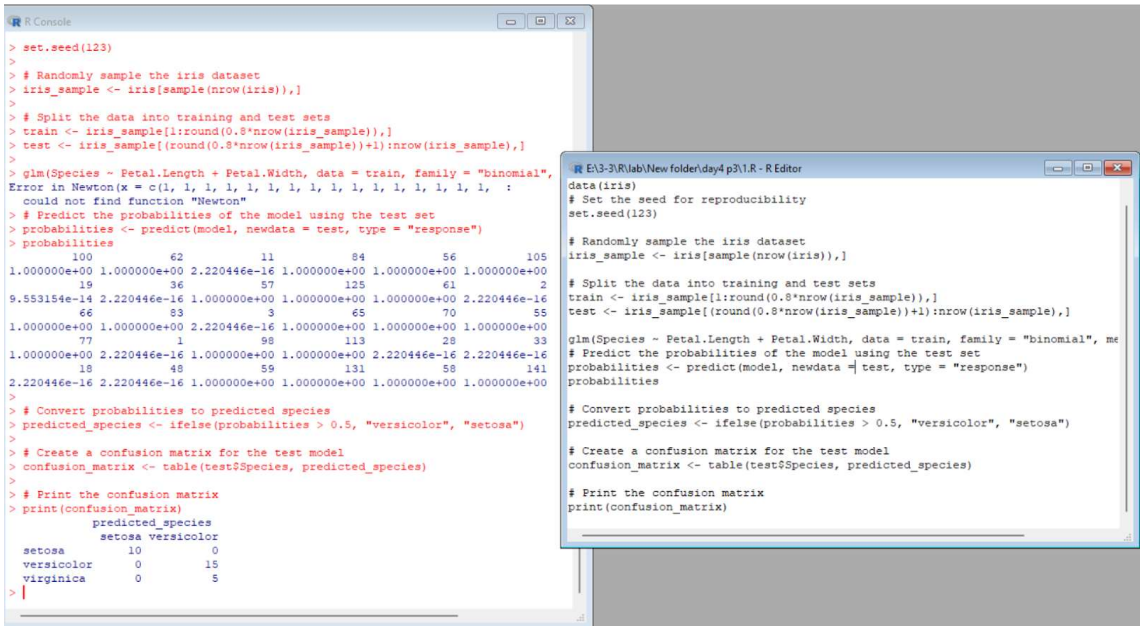
Name: Sreeja Yennam

1.Randomly Sample the iris dataset such as 80% data for training and 20% for test and create Logistics regression with train data, use species as target and petals width and length as feature variables , Predict the probability of the model using test data, Create Confusion matrix for above test model.

SOURCE CODE:

```
# Load the iris dataset
data(iris)
# Set the seed for reproducibility
set.seed(123)

# Randomly sample the iris dataset
iris_sample <- iris[sample(nrow(iris)),]

# Split the data into training and test sets
train <- iris_sample[1:round(0.8*nrow(iris_sample)),]
test <- iris_sample[(round(0.8*nrow(iris_sample))+1):nrow(iris_sample),]

glm(Species ~ Petal.Length + Petal.Width, data = train, family = "binomial", method =
"Newton")
# Predict the probabilities of the model using the test set
probabilities <- predict(model, newdata = test, type = "response")
probabilities

# Convert probabilities to predicted species
predicted_species <- ifelse(probabilities > 0.5, "versicolor", "setosa")
```

```
# Create a confusion matrix for the test model
confusion_matrix <- table(test$Species, predicted_species)

# Print the confusion matrix
print(confusion_matrix)
```



2. (i)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)

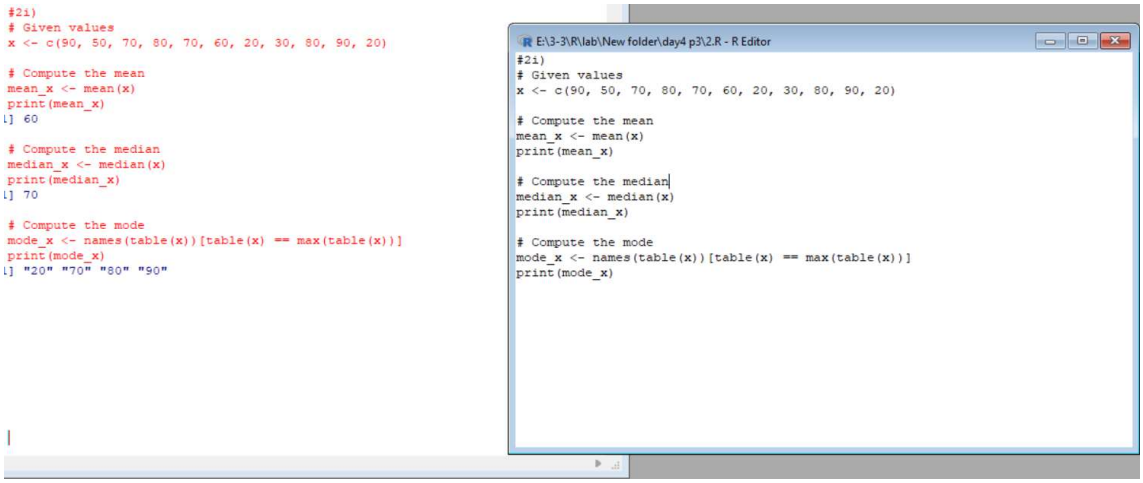
(ii) Write R code to find 2nd highest and 3rd Lowest value of above problem.

SOURCE CODE:

```
#2i)
# Given values
x <- c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)

# Compute the mean
mean_x <- mean(x)
print(mean_x)

# Compute the median
median_x <- median(x)
```

print(median_x)

# Compute the mode
mode_x <- names(table(x))[table(x) == max(table(x))]
print(mode_x)



2II)

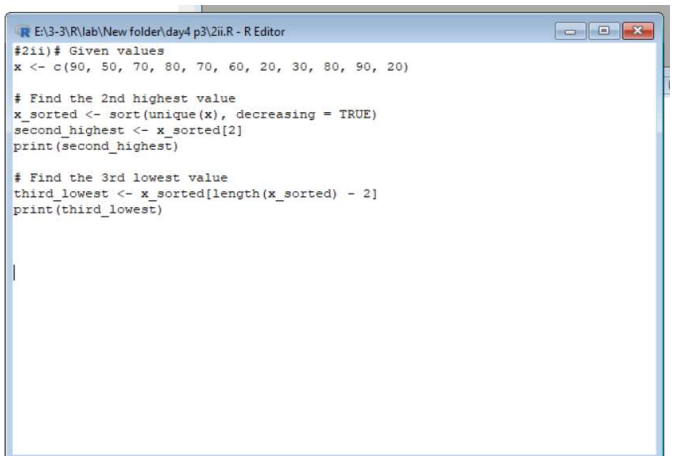
SOURCE CODE:

#2ii)# Given values
x <- c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)

# Find the 2nd highest value
x_sorted <- sort(unique(x), decreasing = TRUE)
second_highest <- x_sorted[2]
print(second_highest)

# Find the 3rd lowest value
third_lowest <- x_sorted[length(x_sorted) - 2]
print(third_lowest)

```
>
>
>
> #2ii)# Given values
> x <- c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)
>
> # Find the 2nd highest value
> x_sorted <- sort(unique(x), decreasing = TRUE)
> second_highest <- x_sorted[2]
> print(second_highest)
[1] 80
>
> # Find the 3rd lowest value
> third_lowest <- x_sorted[length(x_sorted) - 2]
> print(third_lowest)
[1] 50
>
>
>
>
>
>
>
>
>
>
>
>
```



3. 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 -4).

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?

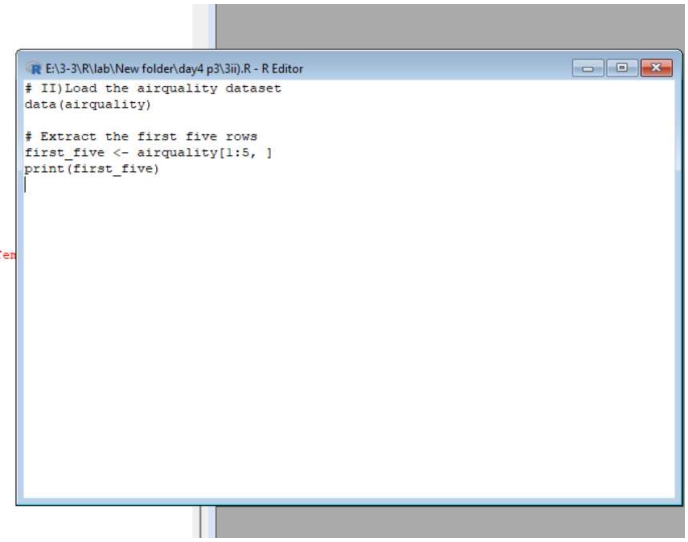
SOURCE CODE:

```
# I)Load the airquality dataset
data(airquality)

# Compute the mean temperature
mean_temp <- sum(airquality$Temp) / length(airquality$Temp)
print(mean_temp)
```

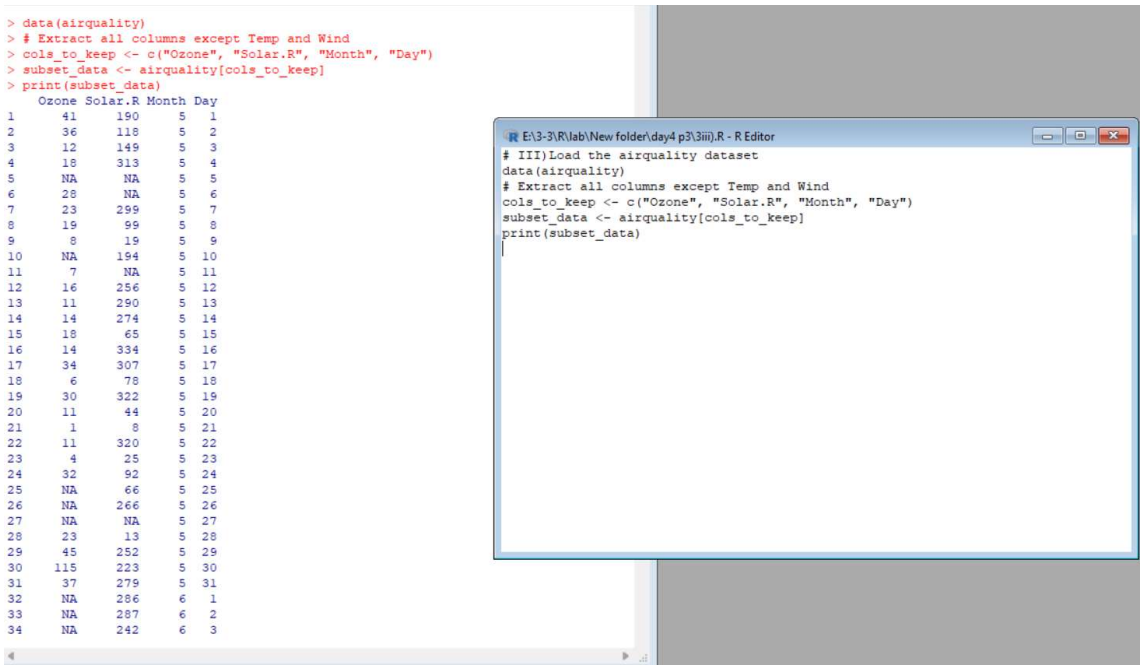
# II)Load the airquality dataset
data(airquality)

# Extract the first five rows
first_five <- airquality[1:5,]
print(first_five)

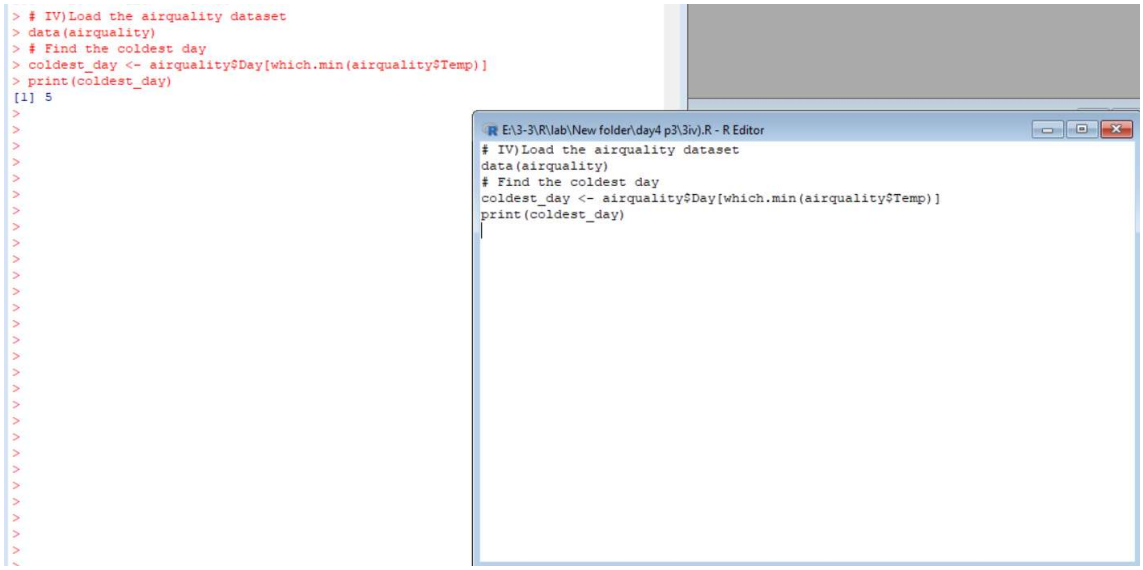
```
# Load the airquality dataset
data(airquality)

# Compute the mean temperature
mean_temp <- sum(airquality$Temp) / length(airquality$Tem
print(mean_temp)
[1] 77.88235


# II)Load the airquality dataset
data(airquality)

# Extract the first five rows
first_five <- airquality[1:5, ]
print(first_five)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
```



# III)Load the airquality dataset
data(airquality)
# Extract all columns except Temp and Wind
cols_to_keep <- c("Ozone", "Solar.R", "Month", "Day")
subset_data <- airquality[cols_to_keep]
print(subset_data)

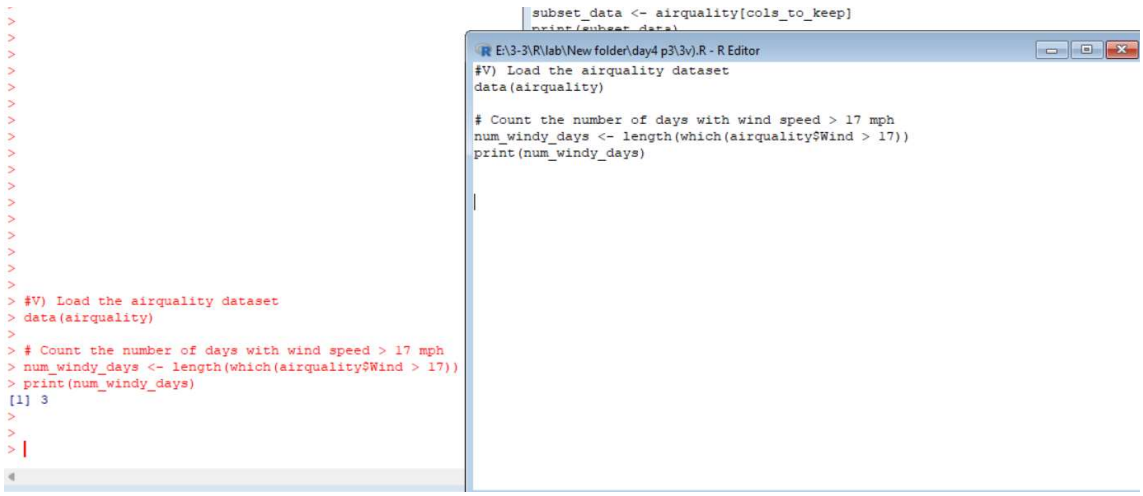


# IV)Load the airquality dataset
data(airquality)
# Find the coldest day
coldest_day <- airquality$Day[which.min(airquality$Temp)]
print(coldest_day)



#V) Load the airquality dataset
data(airquality)

# Count the number of days with wind speed > 17 mph
num_windy_days <- length(which(airquality$Wind > 17))
print(num_windy_days)



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

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

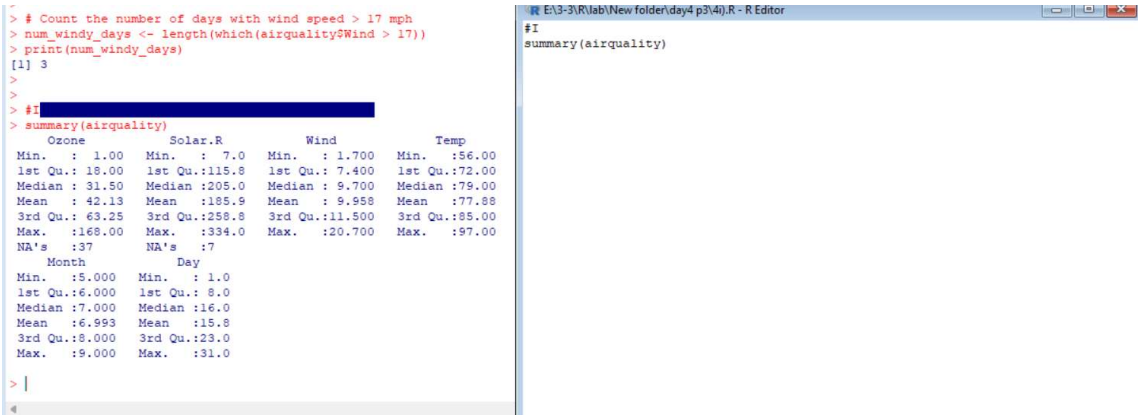
(iii)Melt airquality data and specify month and day to be "ID variables"?

(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:

#i)

summary(airquality)

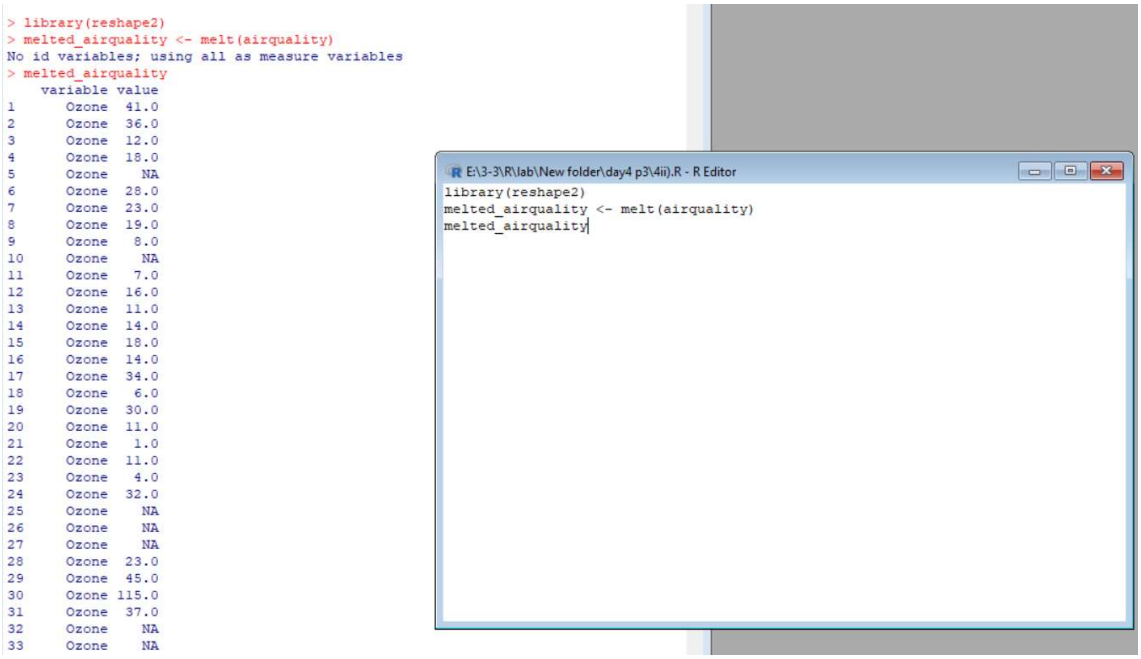


ii)

library(reshape2)

melted_airquality <- melt(airquality)

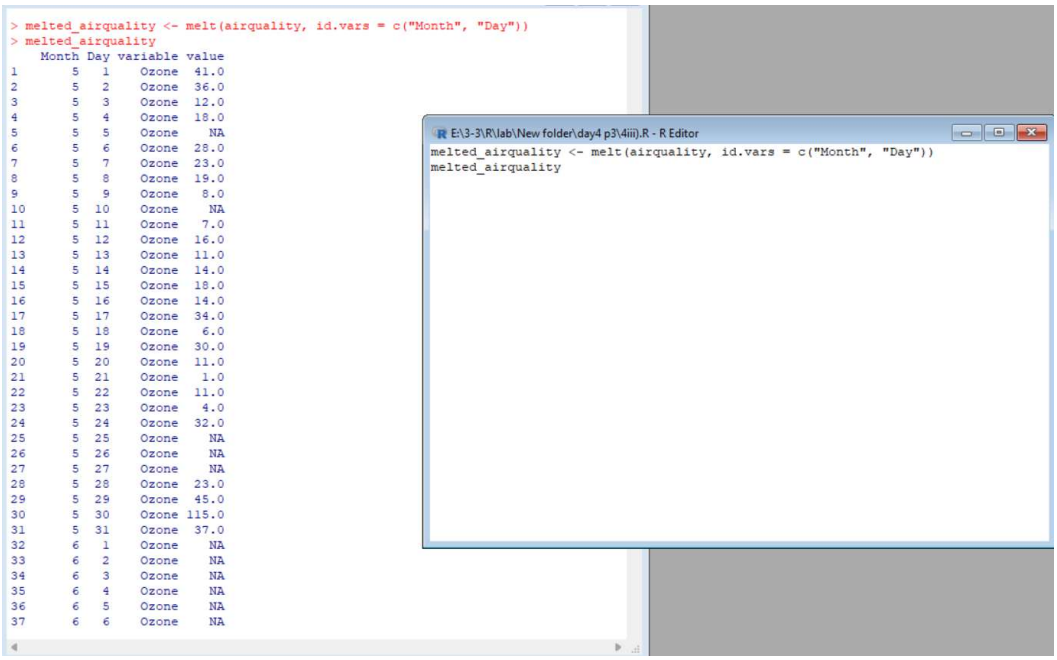
melted_airquality



iii)

melted_airquality <- melt(airquality, id.vars = c("Month", "Day"))

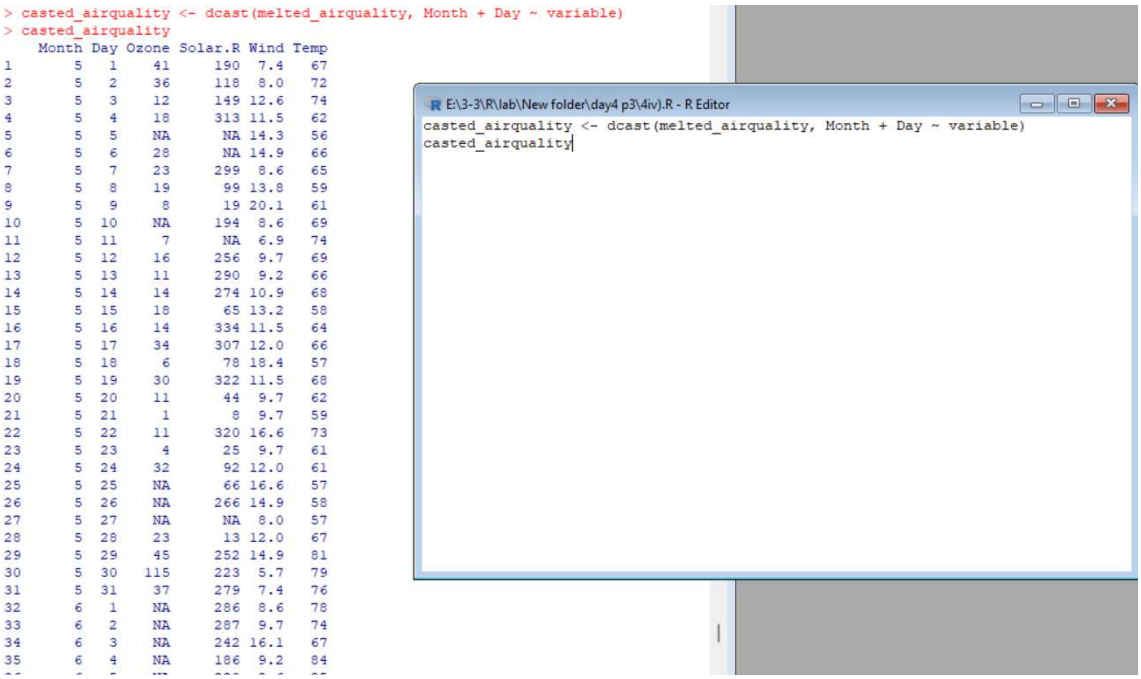
melted_airquality



iv)

casted_airquality <- dcast(melted_airquality, Month + Day ~ variable)

casted_airquality

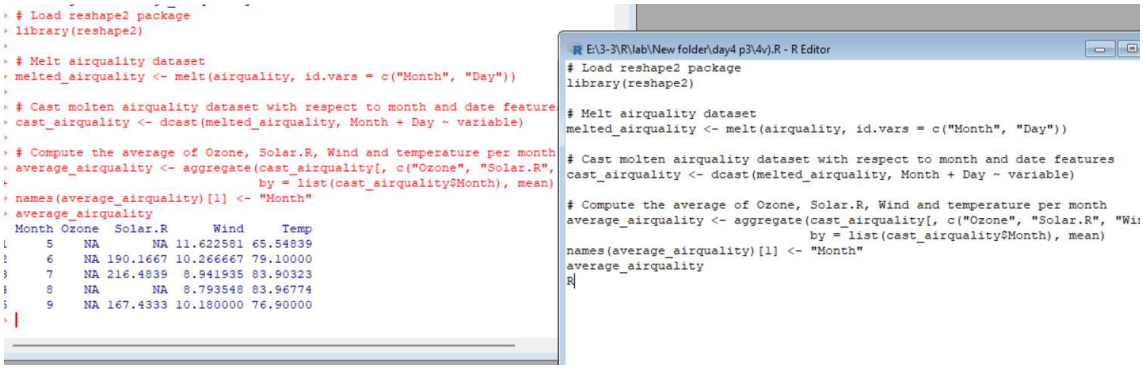


v)

Load reshape2 package

library(reshape2)

```
# Melt airquality dataset
melted_airquality <- melt(airquality, id.vars = c("Month", "Day"))

# Cast molten airquality dataset with respect to month and date features
cast_airquality <- dcast(melted_airquality, Month + Day ~ variable)

# Compute the average of Ozone, Solar.R, Wind and temperature per month
average_airquality <- aggregate(cast_airquality[, c("Ozone", "Solar.R", "Wind", "Temp")],
                                by = list(cast_airquality$Month), mean)
names(average_airquality)[1] <- "Month"
average_airquality
```



5.(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.

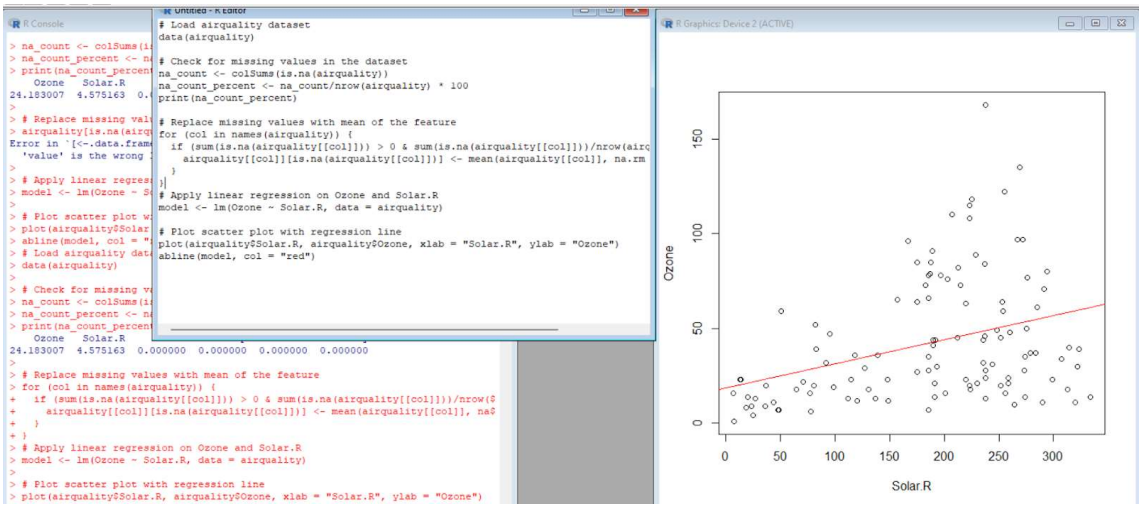
(ii) Apply a linear regression algorithm using Least Squares Method on “Ozone” and “Solar.R”

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

SOURCE CODE:

```
# Load airquality dataset
data(airquality)

# Check for missing values in the dataset
na_count <- colSums(is.na(airquality))
na_count_percent <- na_count/nrow(airquality) * 100
print(na_count_percent)
```

```
# Replace missing values with mean of the feature
for (col in names(airquality)) {
  if (sum(is.na(airquality[[col]])) > 0 & sum(is.na(airquality[[col]]))/nrow(airquality) < 0.1) {
    airquality[[col]][is.na(airquality[[col]])] <- mean(airquality[[col]], na.rm = TRUE)
  }
}
# Apply linear regression on Ozone and Solar.R
model <- lm(Ozone ~ Solar.R, data = airquality)

# Plot scatter plot with regression line
plot(airquality$Solar.R, airquality$Ozone, xlab = "Solar.R", ylab = "Ozone")
abline(model, col = "red")
```



6. Load dataset named ChickWeight,

(i).Order the data frame, in ascending order by feature name “weight” grouped by feature

“diet” and Extract the last 6 records from order data frame.

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

b. Perform cast function to display the mean value of weight grouped by Diet

c. Perform cast function to display the mode of weight grouped by Diet

SOURCE CODE:

```
# Load ChickWeight dataset
data(ChickWeight)

# (i) Order the data frame, in ascending order by feature name “weight” grouped by feature
“diet” and extract the last 6 records
```

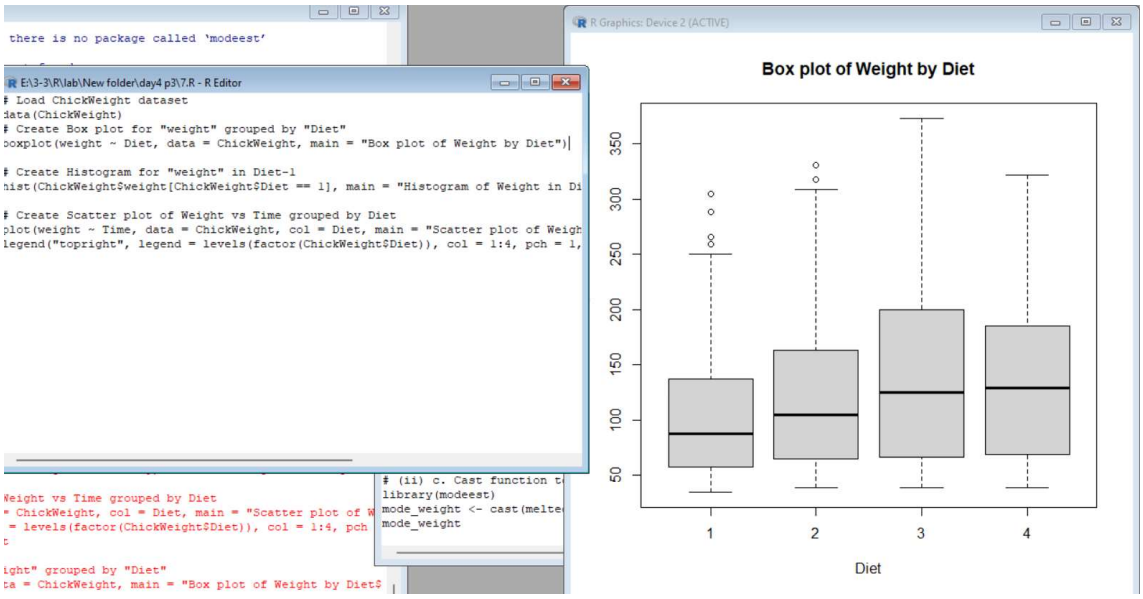
```
ordered_data <- ChickWeight[order(ChickWeight$weight), ]
last_6 <- tail(ordered_data, 6)

# (ii) Melt and cast functions
library(reshape2)

# (ii) a. Melt function
melted_chickweight <- melt(ChickWeight, id.vars = c("Chick", "Time", "Diet"))

# (ii) b. Cast function to display the mean value of weight grouped by Diet
mean_weight <- cast(melted_chickweight, Diet ~ ., mean)

# (ii) c. Cast function to display the mode of weight grouped by Diet
library(modeest)
mode_weight <- cast(melted_chickweight, Diet ~ ., modeest::mfv)
```

7. a. Create Box plot for “weight” grouped by “Diet”
   b. Create a Histogram for “weight” features belong to Diet- 1 category
   c. Create Scatter plot for “ weight” vs “Time” grouped by Diet

SOURCE CODE:

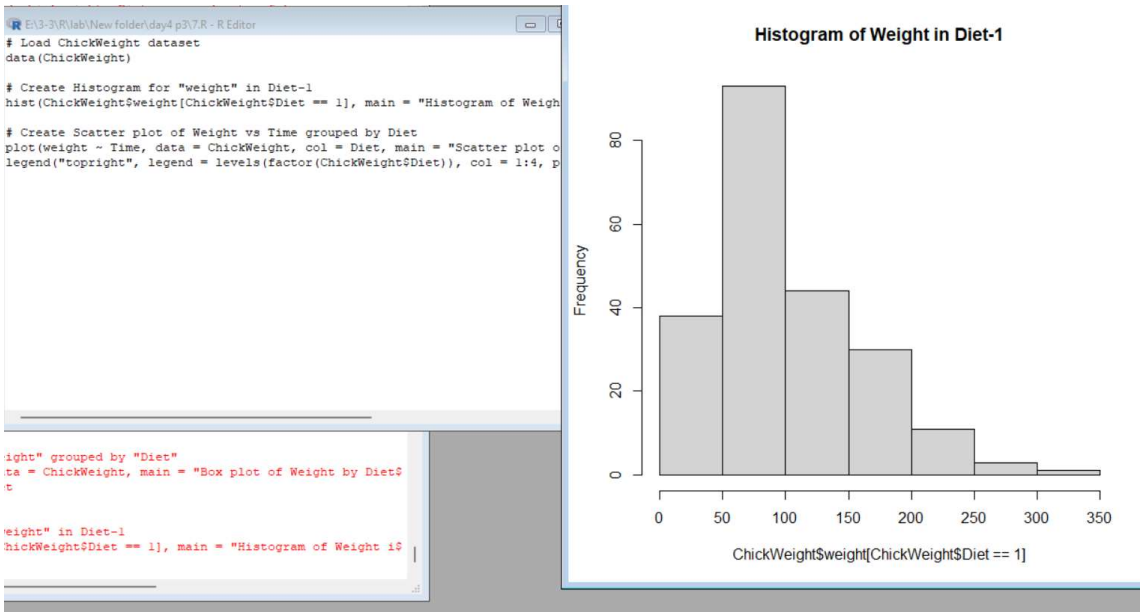
A.

```
# Load ChickWeight dataset
data(ChickWeight)
# Create Box plot for "weight" grouped by "Diet"
boxplot(weight ~ Diet, data = ChickWeight, main = "Box plot of Weight by Diet")
```



b.

```
# Load ChickWeight dataset
data(ChickWeight)

# Create Histogram for "weight" in Diet-1
hist(ChickWeight$weight[ChickWeight$Diet == 1], main = "Histogram of Weight in Diet-1")
```



c.

```
# Load ChickWeight dataset
```

data(ChickWeight)

Create Scatter plot of Weight vs Time grouped by Diet

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

legend("topright", legend = levels(factor(ChickWeight$Diet)), col = 1:4, pch = 1, title = "Diet")