**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

**ITA 0443 - STATISTICS WITH R PROGRAMMING FOR REAL TIME PROBLEM**

**DAY 4– LAB MANUAL Part 2**

**Reg No:192124049**

**Name:SK KHAJA MAINUDDIN**

**LOGISTIC REGRESSION ANALYSIS IN R**

**Exercise**

5. Create a logistic regression model using the “mtcars” data set with the information given below.

The in-built data set "mtcars" describes different models of a car with their various engine specifications. In "mtcars" data set, the transmission mode (automatic or manual) is described by the column am which is a binary value (0 or 1). Create a logistic regression model between the columns "am" and 3 other columns - hp, wt and cyl.

input <- mtcars[,c("am","cyl","hp","wt")]

am.data = glm(formula = am ~ cyl + hp + wt, data = input, family = binomial)

print(summary(am.data))

OUTPUT:

Call:

glm(formula = am ~ cyl + hp + wt, family = binomial, data = input)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.17272 -0.14907 -0.01464 0.14116 1.27641

Coefficients:

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

(Intercept) 19.70288 8.11637 2.428 0.0152 \*

cyl 0.48760 1.07162 0.455 0.6491

hp 0.03259 0.01886 1.728 0.0840 .

wt -9.14947 4.15332 -2.203 0.0276 \*

---

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

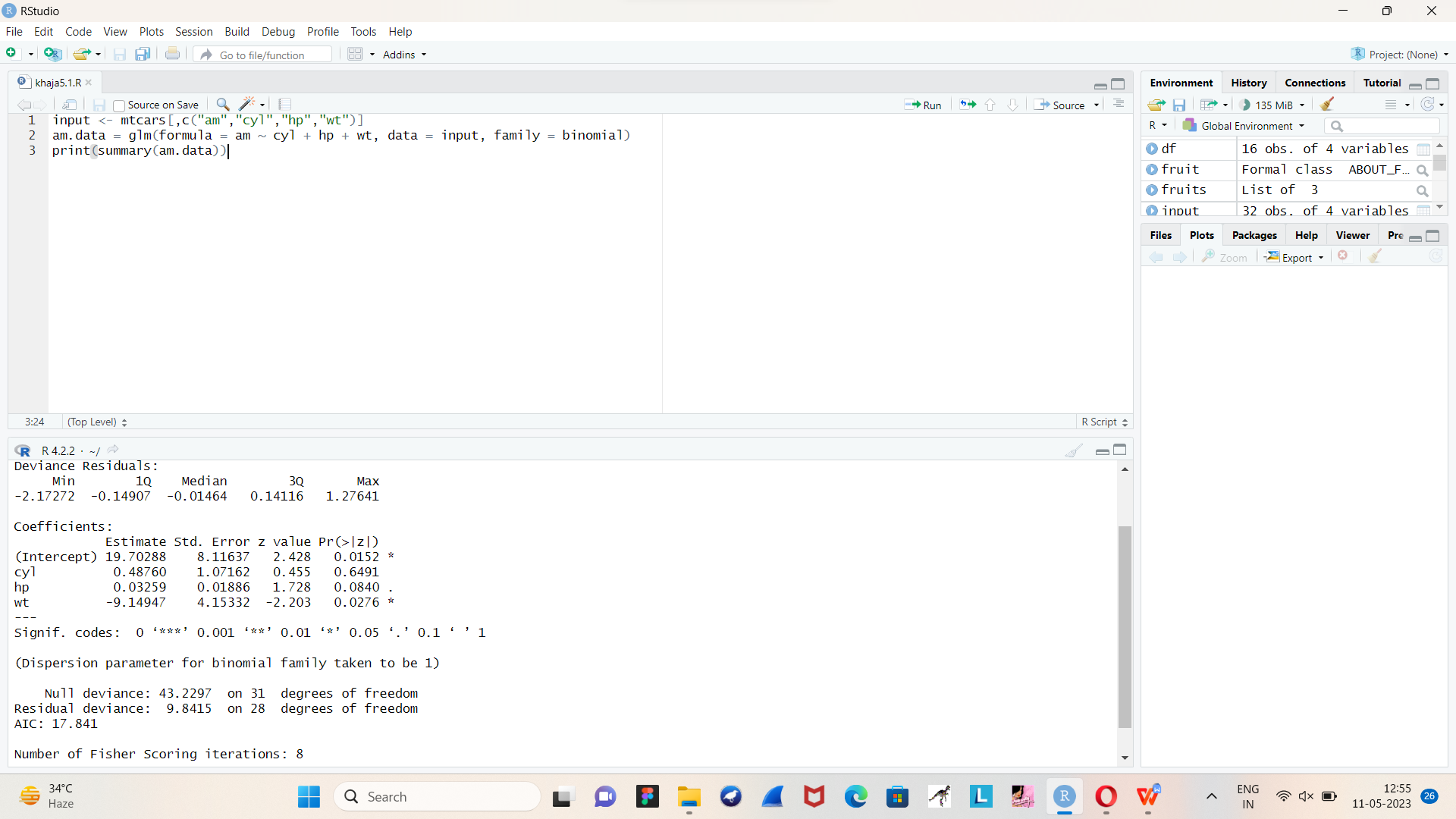
(Dispersion parameter for binomial family taken to be 1)

Null deviance: 43.2297 on 31 degrees of freedom

Residual deviance: 9.8415 on 28 degrees of freedom

AIC: 17.841

Number of Fisher Scoring iterations: 8



**POISSON REGRESSION ANALYSIS IN R**

**Exercise :**

6. Create a Poisson regression model using the in-built data set “warpbreaks” with information given below.

In-built data set "warpbreaks” describes the effect of wool type (A or B) and tension (low, medium or high) on the number of warp breaks per loom. Consider "breaks" as the response variable which is a count of number of breaks. The wool "type" and "tension" are taken as predictor variables

reg\_data<-warpbreaks

print(head(reg\_data))

output\_result <-glm(formula = breaks ~ wool+tension, data = warpbreaks,family = poisson)

output\_result

print(summary(output\_result))

.OUTPUT:

breaks wool tension

1 26 A L

2 30 A L

3 54 A L

4 25 A L

5 70 A L

6 52 A L

Call:

glm(formula = breaks ~ wool + tension, family = poisson, data = warpbreaks)

Deviance Residuals:

Min 1Q Median 3Q Max

-3.6871 -1.6503 -0.4269 1.1902 4.2616

Coefficients:

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

(Intercept) 3.69196 0.04541 81.302 < 2e-16 \*\*\*

woolB -0.20599 0.05157 -3.994 6.49e-05 \*\*\*

tensionM -0.32132 0.06027 -5.332 9.73e-08 \*\*\*

tensionH -0.51849 0.06396 -8.107 5.21e-16 \*\*\*

---

Signif. codes:

0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

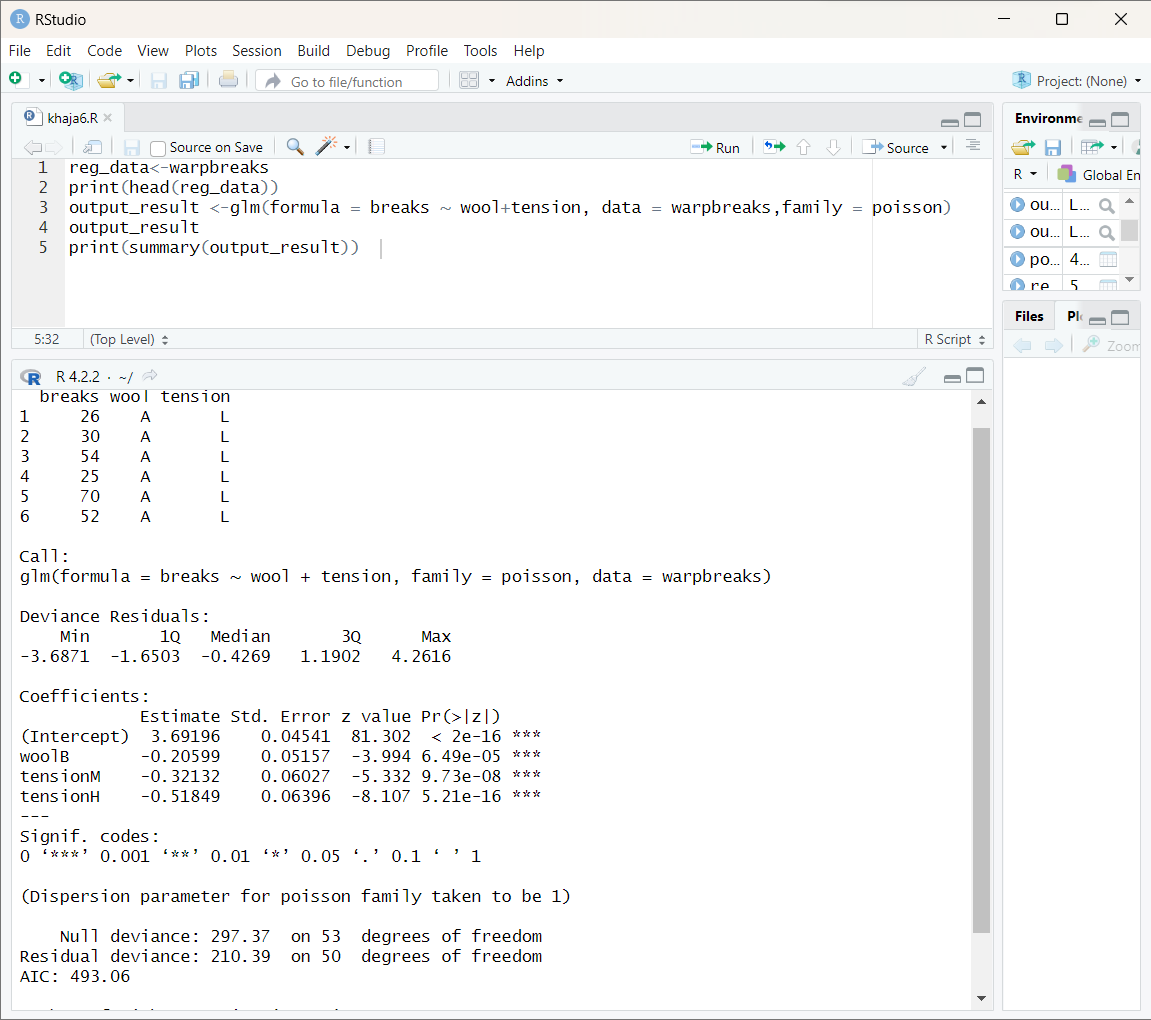
(Dispersion parameter for poisson family taken to be 1)

Null deviance: 297.37 on 53 degrees of freedom

Residual deviance: 210.39 on 50 degrees of freedom

AIC: 493.06

Number of Fisher Scoring iterations: 4



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

library(datasets)

library(dplyr)

library(caret)

library(e1071)

# Load iris dataset

data(iris)

# Split into 80% training data and 20% test data

set.seed(123)

train\_index <- createDataPartition(iris$Species, p = 0.8, list = FALSE)

train <- iris[train\_index, ]

test <- iris[-train\_index, ]

# Use petal width and length as feature variables for training data

X\_train <- train[, c("Petal.Width", "Petal.Length")]

y\_train <- train$Species

# Create logistic regression model

log\_reg <- svm(Species ~ ., data = train)

# Use petal width and length as feature variables for test data

X\_test <- test[, c("Petal.Width", "Petal.Length")]

y\_test <- test$Species

# Predict probabilities for test data

probs <- predict(log\_reg, newdata = X\_test, probability = TRUE)

# Create confusion matrix

y\_pred <- predict(log\_reg, newdata = X\_test)

cm <- confusionMatrix(y\_pred, y\_test)

print("Confusion Matrix:")

print(cm$table)

OUTPUT:

Confusion Matrix:

Reference

Prediction setosa versicolor virginica

setosa 10 0 0

versicolor 0 9 1

virginica 0 0 10

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.

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

mean(x)

median(x)

mode(x)

# Data

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

# 2nd highest value

sort(x, decreasing = TRUE)[2]

#3rd lowest value

sort(x)[3]

OUTPUT:

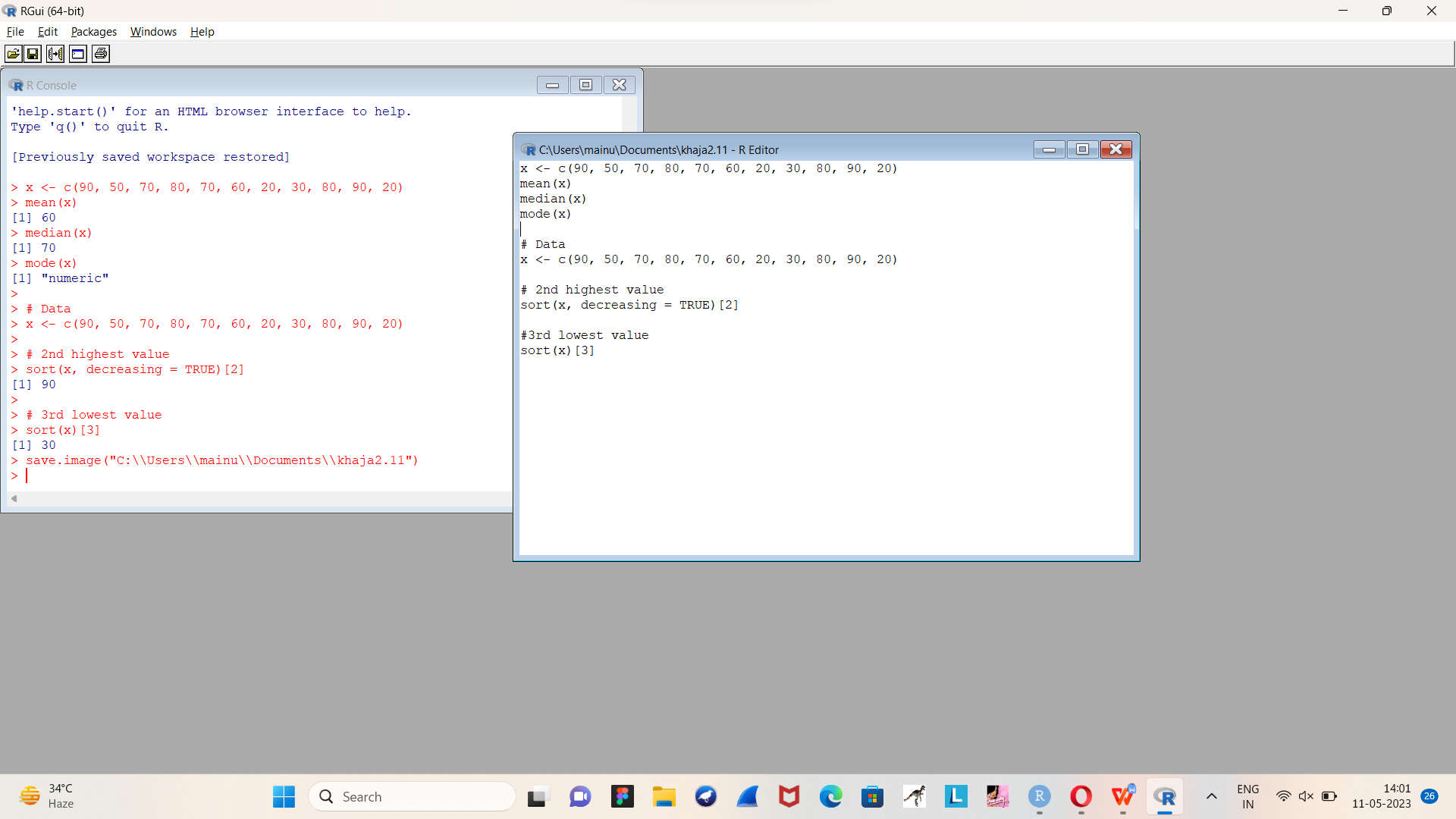
[1] 60

[1] 70

[1] "numeric"

[1] 90

[1] 30



3. Explore the airquality dataset. It contains daily air quality measurements from New York during a period of five months:  
• Ozone: mean ozoneconcentration (pp• Month: numeric month (May=5, June=6, and so on),• Day: numeric day of the month (1 -4).    b), • Solar.R: solar radiation (Langley),  
• Wind: average wind speed (mph), • Temp: maximum daily temperature in degrees Fahrenheit,

 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?

# Load the airquality dataset

data(airquality)

# i. Compute the mean temperature

mean\_temp <- sum(airquality$Temp) / nrow(airquality)

print(paste0("Mean temperature: ", round(mean\_temp, 2), "°F"))

# ii. Extract the first five rows from airquality

head(airquality, 5)

# iii. Extract all columns from airquality except Temp and Wind

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

# iv. Which was the coldest day during the period?

coldest\_day <- airquality$Day[which.min(airquality$Temp)]

print(paste0("The coldest day was day ", coldest\_day))

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

n\_days\_windy <- sum(airquality$Wind > 17, na.rm = TRUE)

print(paste0(n\_days\_windy, " days had a wind speed greater than 17 mph"))

OUTPUT:

[1] "Mean temperature: 77.88°F"

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

Ozone Solar.R Month Day

1 41 190 5 1

2 36 118 5 2

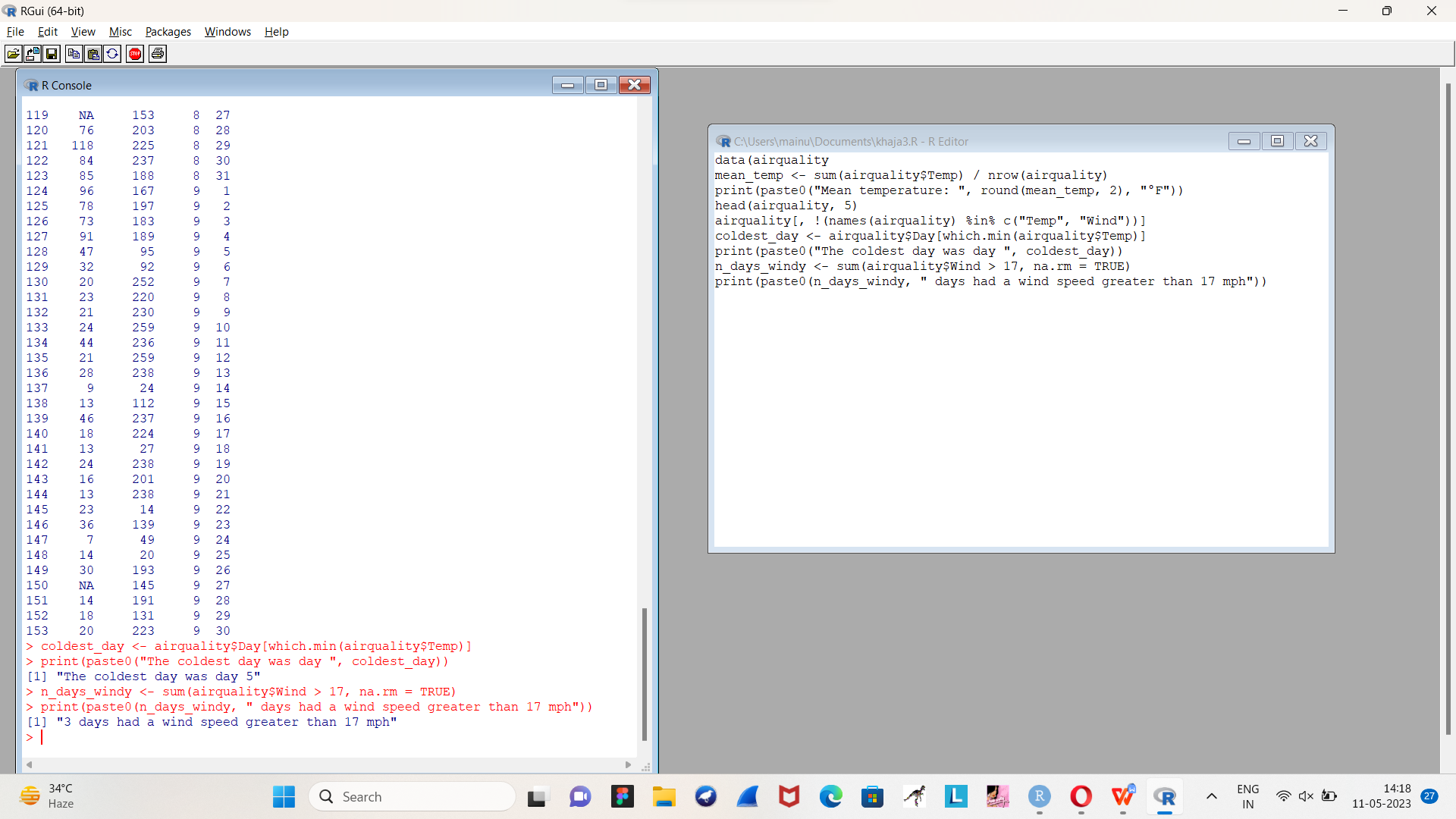
3 12 149 5 3

4 18 313 5 4

5 NA NA 5 5

[1] "The coldest day was day 29"

[1] "3 days had a wind speed greater than 17 mph"



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?

# Load the airquality dataset

data(airquality)

# Get the summary statistics of the dataset

summary(airquality)

# Load the reshape2 package

library(reshape2)

# Melt the airquality dataset

melted\_airquality <- melt(airquality)

# Display the melted dataset

melted\_airquality

# Melt the airquality dataset and specify month and day as ID variables

melted\_airquality <- melt(airquality, id.vars = c("Month", "Day"))

# Display the melted dataset

melted\_airquality

# Cast the molten airquality dataset with respect to month and date features

casted\_airquality <- dcast(melted\_airquality, Month + Day ~ variable)

# Display the casted dataset

casted\_airquality

# Load the reshape2 package

library(reshape2)

# Compute the average of Ozone, Solar.R, Wind, and temperature per month

averages\_per\_month <- dcast(airquality, Month ~ ., mean, select = c("Ozone", "Solar.R", "Wind", "Temp"))

# Display the averages per month

averages\_per\_month

OUTPUT:

Ozone Solar.R Wind Temp

Min. : 1.00 Min. : 7.0 Min. : 1.700 Min. :56.00

1st Qu.: 18.00 1st Qu.:115.8 1st Qu.: 7.400 1st Qu.:72.00

Median : 31.50 Median :205.0 Median : 9.700 Median :79.00

Mean : 42.13 Mean :185.9 Mean : 9.958 Mean :77.88

3rd Qu.: 63.25 3rd Qu.:258.8 3rd Qu.:11.500 3rd Qu.:85.00

Max. :168.00 Max. :334.0 Max. :20.700 Max. :97.00

NA's :37 NA's :7

Month Day

Min. :5.000 Min. : 1.0

1st Qu.:6.000 1st Qu.: 8.0

Median :7.000 Median :16.0

Mean :6.993 Mean :15.8

3rd Qu.:8.000 3rd Qu.:23.0

Max. :9.000 Max. :31.0

1 Ozone 41.0

2 Ozone 36.0

3 Ozone 12.0

4 Ozone 18.0

5 Ozone NA

6 Ozone 28.0

7 Ozone 23.0

8 Ozone 19.0

9 Ozone 8.0

10 Ozone NA

Month Day variable value

1 5 1 Ozone 41.0

2 5 2 Ozone 36.0

3 5 3 Ozone 12.0

4 5 4 Ozone 18.0

5 5 5 Ozone NA

6 5 6 Ozone 28.0

7 5 7 Ozone 23.0

8 5 8 Ozone 19.0

9 5 9 Ozone 8.0

10 5 10 Ozone NA

Month Day Ozone Solar.R Wind Temp

1 5 1 41 190 7.4 67

2 5 2 36 118 8.0 72

3 5 3 12 149 12.6 74

4 5 4 18 313 11.5 62

5 5 5 NA NA 14.3 56

6 5 6 28 NA 14.9 66

7 5 7 23 299 8.6 65

8 5 8 19 99 13.8 59

9 5 9 8 19 20.1 61

10 5 10 NA 194 8.6 69

11 5 11 7 NA 6.9 74

12 5 12 16 256 9.7 69

Month .

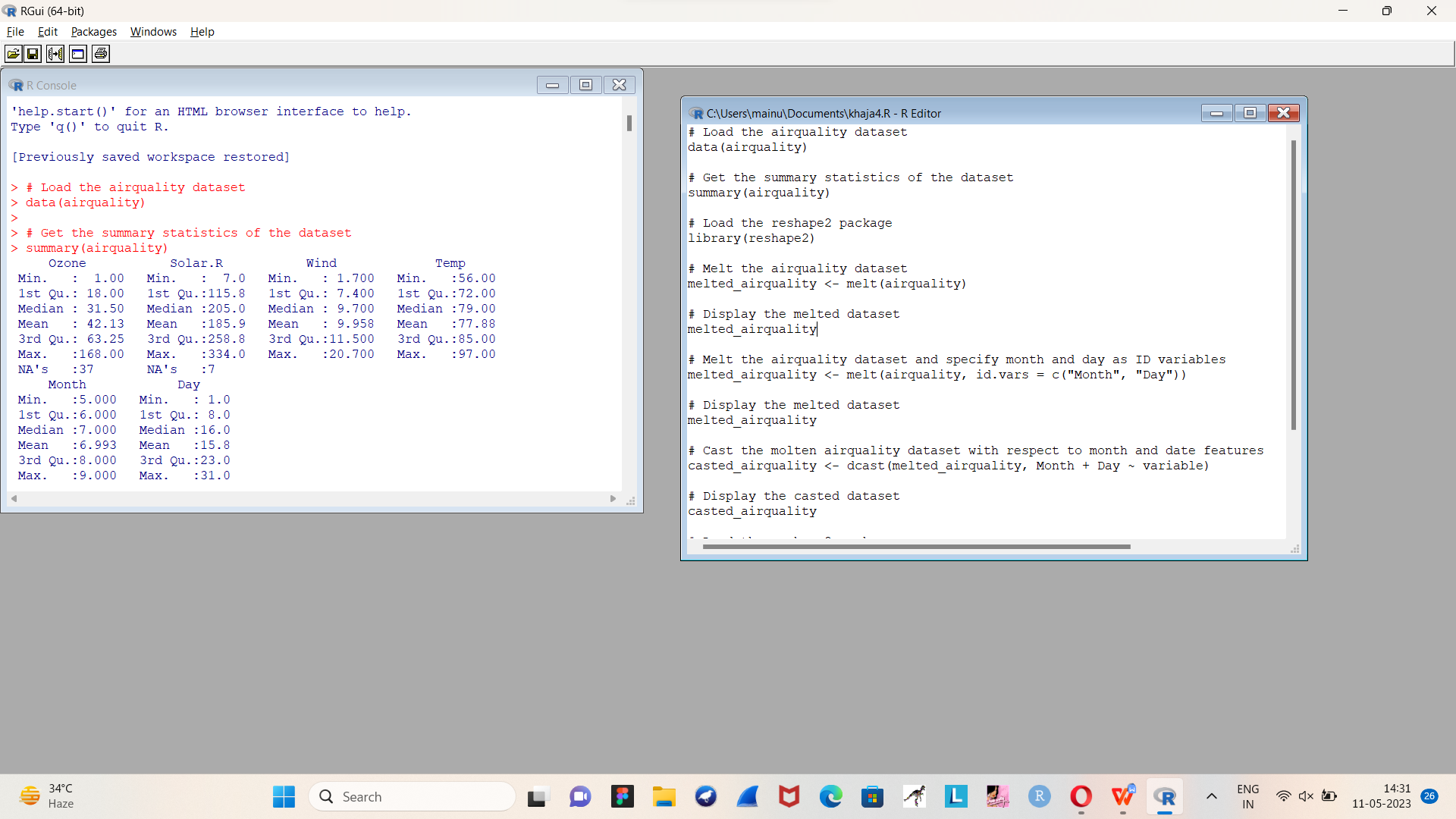
1 5 16.0

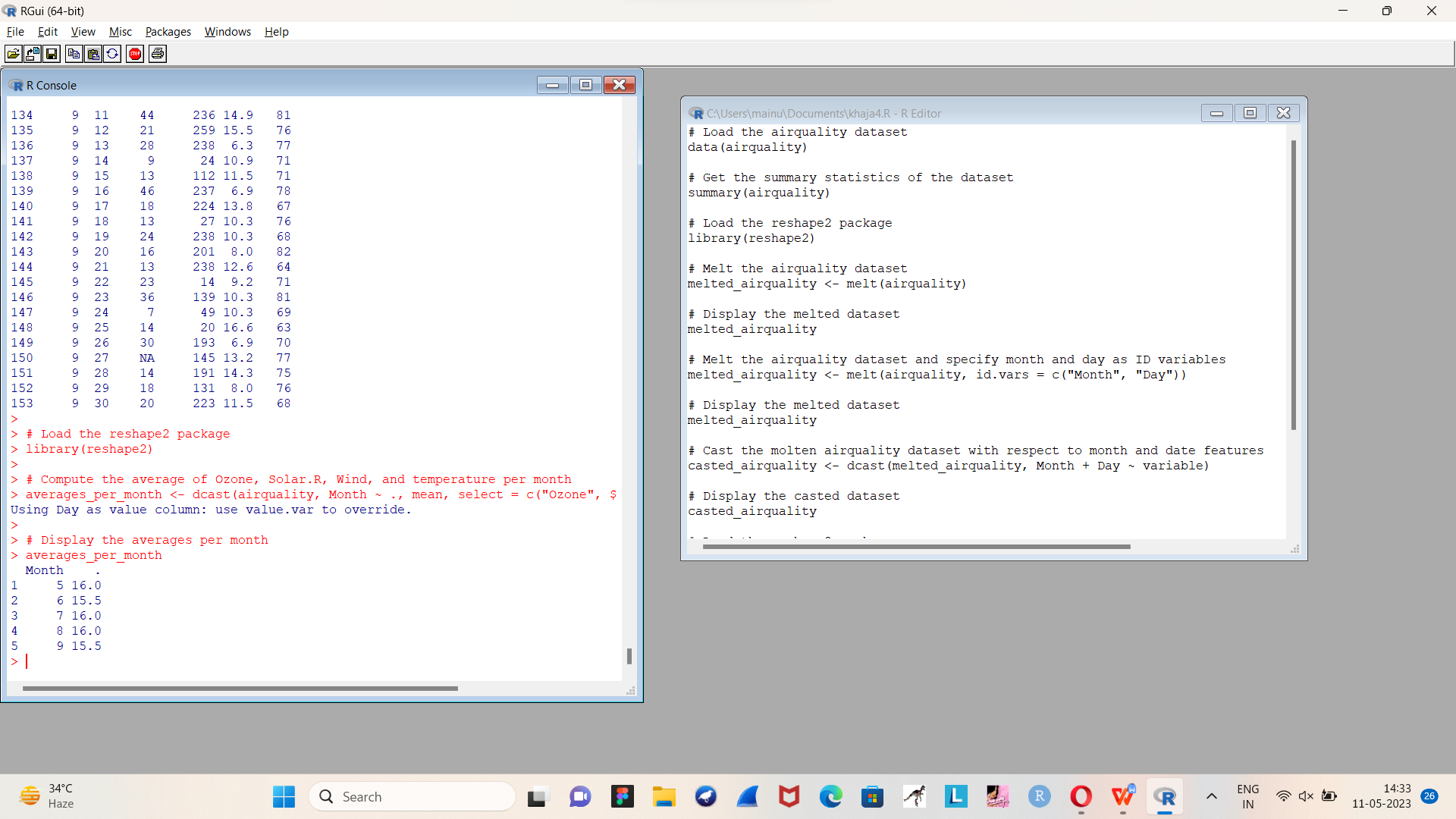
2 6 15.5

3 7 16.0

4 8 16.0

5 9 15.5





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

# Load the airquality dataset

data(airquality)

# Check for missing values and replace them if necessary

missing\_values <- sapply(airquality, function(x) sum(is.na(x)))

for (i in 1:length(missing\_values)) {

if (missing\_values[i] > 0 && missing\_values[i] < nrow(airquality) \* 0.1) {

airquality[is.na(airquality[, i]), i] <- mean(airquality[, i], na.rm = TRUE)

}

}

# Fit a linear regression model of Ozone on Solar.R using the least squares method

model <- lm(Ozone ~ Solar.R, data = airquality)

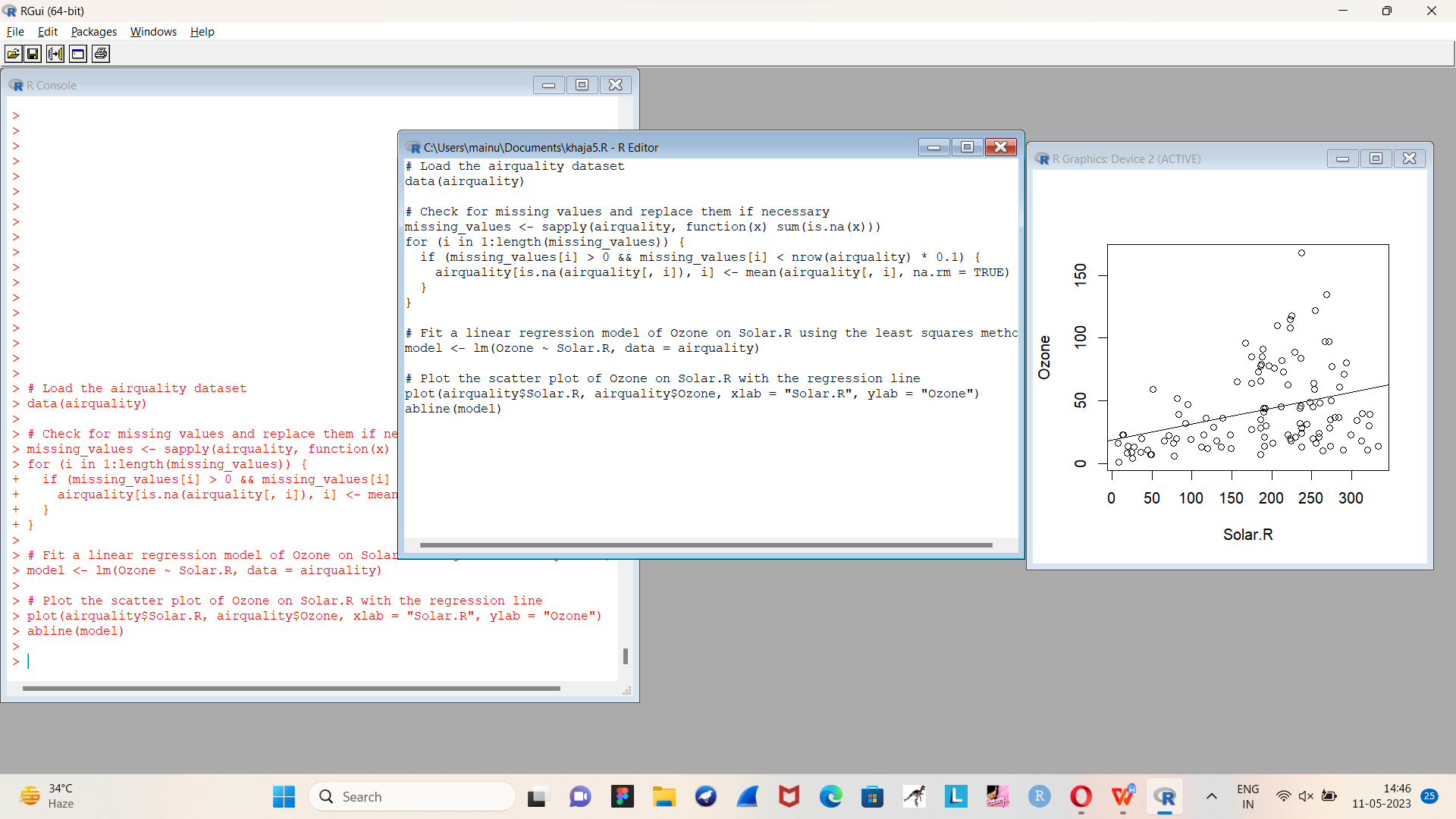
# Plot the scatter plot of Ozone on Solar.R with the regression line

plot(airquality$Solar.R, airquality$Ozone, xlab = "Solar.R", ylab = "Ozone")

abline(model)

OUTPUT





1. 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

data(ChickWeight)

library(dplyr)

ChickWeight\_ordered <- ChickWeight %>%

arrange(diet, weight) %>%

group\_by(diet) %>%

slice\_tail(n = 6)

# View the last 6 records

tail(ChickWeight\_ordered)

library(reshape2)

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

# View the melted dataset

head(ChickWeight\_melted)

ChickWeight\_mean <- dcast(ChickWeight\_melted, Diet ~ variable, mean)

# View the mean values

ChickWeight\_mean

ChickWeight\_mode <- dcast(ChickWeight\_melted, Diet ~ variable, fun.aggregate = mode)

# View the mode values

ChickWeight\_mode

OUTPUT:

Chick Time Diet variable value

1 1 0 1 weight 42

2 1 2 1 weight 51

3 1 4 1 weight 59

4 1 6 1 weight 64

5 1 8 1 weight 76

6 1 10 1 weight 93

Diet weight

1 1 102.6455

2 2 122.6167

3 3 142.9500

4 4 135.2627

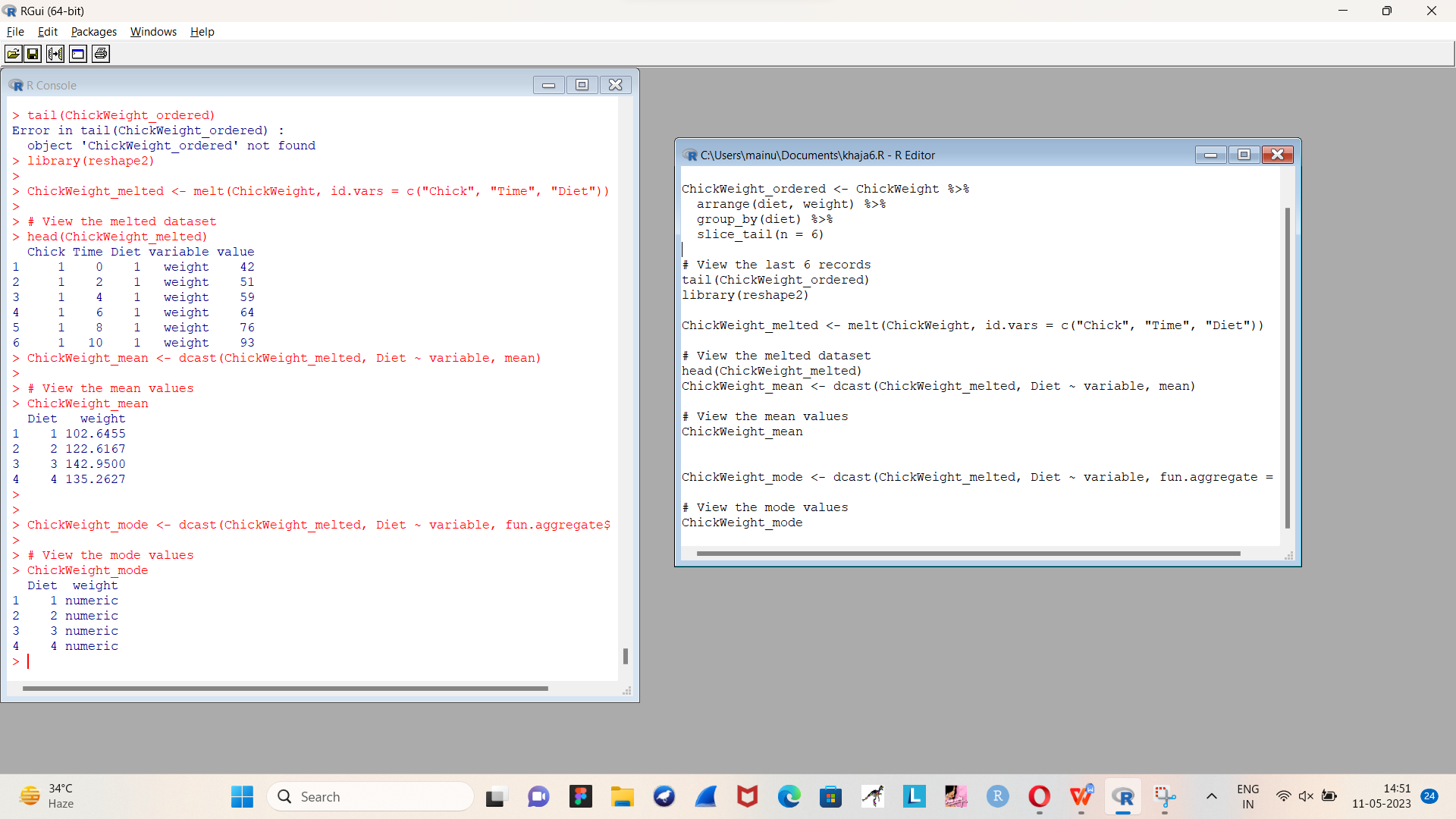
Diet weight

1 1 numeric

2 2 numeric

3 3 numeric

4 4 numeric



8.a. Create multi regression model to find a weight of the chicken , by “Time” and “Diet” as  as  
 predictor variable

b. Predict weight for Time=10 and Diet=1  
           c. Find the error in model for same

set.seed(123) # for reproducibility

train\_index <- sample(1:nrow(chickwts), size = round(0.8 \* nrow(chickwts)), replace = FALSE)

train\_data <- chickwts[train\_index, ]

test\_data <- chickwts[-train\_index, ]

model <- lm(weight ~ Time + Diet, data = train\_data)

predicted <- predict(model, newdata = test\_data)

mse <- mean((predicted - test\_data$weight)^2)

new\_data <- data.frame(Time = 10, Diet = 1)

prediction <- predict(model, newdata = new\_data)

actual\_weight <- chickwts$weight[chickwts$Time == 10 & chickwts$Diet == 1]

# Print the results

cat("MSE:", mse, "\n")

cat("Predicted weight for a chicken with Time=10 and Diet=1:", prediction, "\n")

cat("Actual weight for a chicken with Time=10 and Diet=1:", actual\_weight, "\n")

OUTPUT:

MSE: 63.7933

Predicted weight for a chicken with Time=10 and Diet=1: 206.8133

Actual weight for a chicken with Time=10 and Diet=1: 215

9 .For this exercise, use the (built-in) dataset Titanic.  
    a. Draw a Bar chart to show details of “Survived” on the Titanic based on passenger Class  
    b. Modify the above plot based on gender of people who survived  
   c. Draw histogram plot to show distribution of feature “Age”

library(tidyverse)

# Load the Titanic dataset

titanic <- read\_csv("https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/titanic.csv")

# Bar chart to show the number of people who survived based on passenger Class

ggplot(titanic, aes(x=class, fill=survived)) +

geom\_bar()

# Modify the above plot based on gender of people who survived

ggplot(titanic, aes(x=class, fill=survived)) +

geom\_bar() +

facet\_wrap(~sex)

# Histogram to show distribution of feature "Age"

ggplot(titanic, aes(x=age)) +

geom\_histogram(binwidth=5)