ITA 0451 - STATISTICS WITH R PROGRAMMING

DAY 4 – LAB ASSESSMENT Part 3

Reg No:192121147

Name:P.BARATH

**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**

INPUT:

# Load the iris dataset

library(datasets)

data(iris)

# Set the seed for reproducibility

set.seed(123)

# Randomly sample the iris dataset

train\_index <- sample(nrow(iris), floor(0.8\*nrow(iris)), replace = FALSE)

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

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

# Train a logistic regression model

model <- glm(Species ~ Petal.Width + Petal.Length, data = train\_data, family = "binomial")

# Predict the probabilities using the test data

probabilities <- predict(model, newdata = test\_data, type = "response")

# Convert probabilities to predicted classes

predicted\_classes <- ifelse(probabilities > 0.5, "versicolor", "setosa")

# Create confusion matrix

table(test\_data$Species, predicted\_classes)

OUTPUT:

> # Load the iris dataset

> library(datasets)

> data(iris)

>

> # Set the seed for reproducibility

> set.seed(123)

>

> # Randomly sample the iris dataset

> train\_index <- sample(nrow(iris), floor(0.8\*nrow(iris)), replace = FALSE)

> train\_data <- iris[train\_index, ]

> test\_data <- iris[-train\_index, ]

>

> # Train a logistic regression model

> model <- glm(Species ~ Petal.Width + Petal.Length, data = train\_data, family = "binomial")

>

> # Predict the probabilities using the test data

> probabilities <- predict(model, newdata = test\_data, type = "response")

>

> # Convert probabilities to predicted classes

> predicted\_classes <- ifelse(probabilities > 0.5, "versicolor", "setosa")

>

> # Create confusion matrix

> table(test\_data$Species, predicted\_classes)

predicted\_classes

setosa versicolor

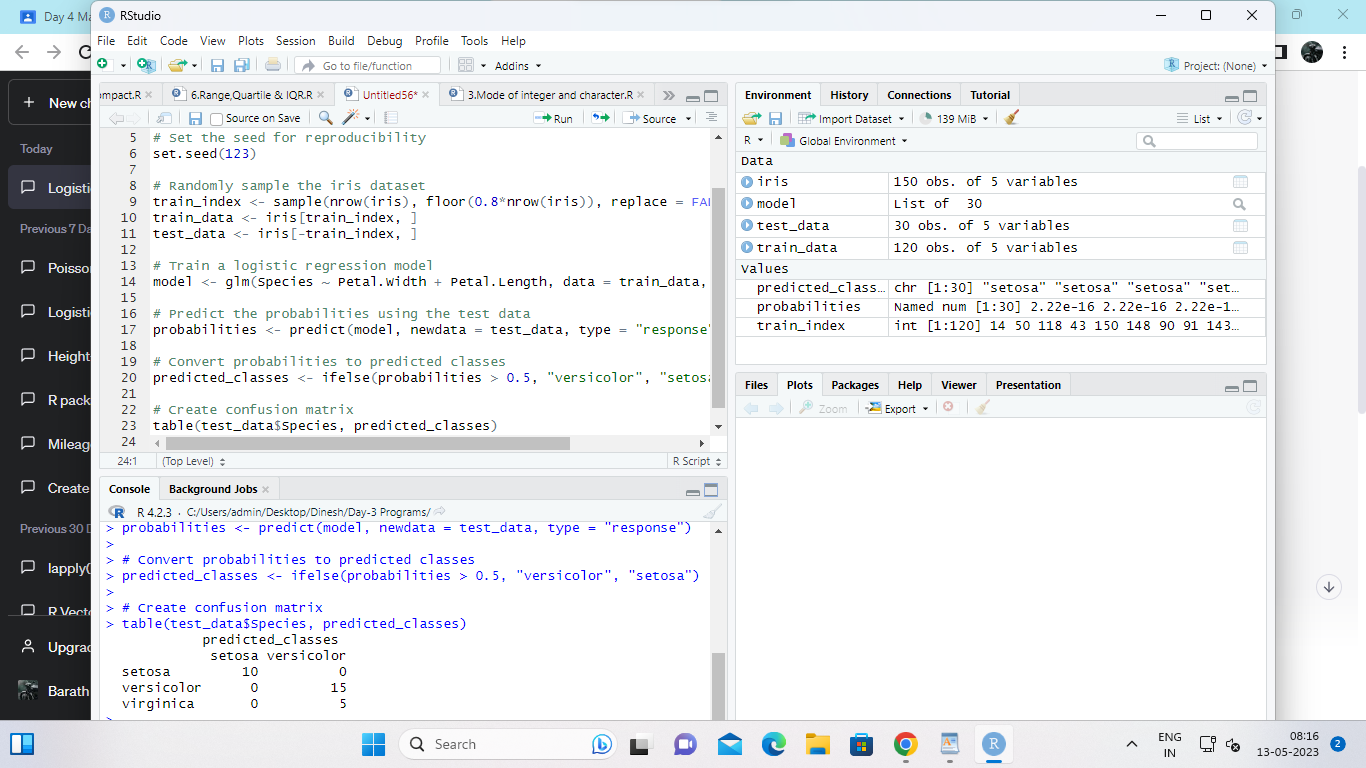
setosa 10 0

versicolor 0 15

virginica 0 5

>

>



**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**)

INPUT:

# Define the values

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

# Compute the mean

mean\_value <- mean(values)

cat("Mean:", mean\_value, "\n")

# Compute the median

median\_value <- median(values)

cat("Median:", median\_value, "\n")

# Compute the mode

mode\_value <- names(sort(table(values), decreasing = TRUE))[1]

cat("Mode:", mode\_value, "\n")

OUTPUT:

> # Define the values

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

>

> # Compute the mean

> mean\_value <- mean(values)

> cat("Mean:", mean\_value, "\n")

Mean: 60

>

> # Compute the median

> median\_value <- median(values)

> cat("Median:", median\_value, "\n")

Median: 70

>

> # Compute the mode

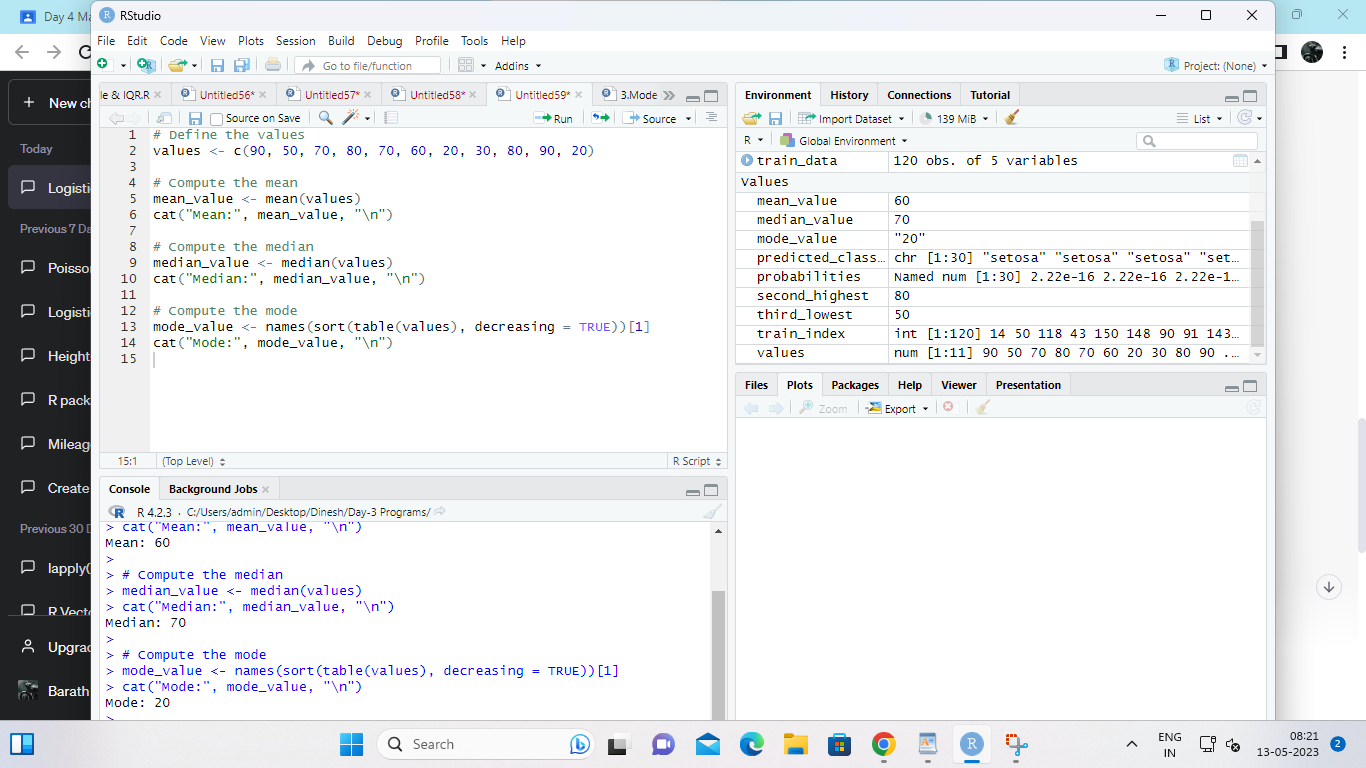
> mode\_value <- names(sort(table(values), decreasing = TRUE))[1]

> cat("Mode:", mode\_value, "\n")

Mode: 20

>

>



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

INPUT:

# Define the values

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

# Find the second highest value

second\_highest <- sort(unique(values), decreasing = TRUE)[2]

cat("Second highest:", second\_highest, "\n")

# Find the third lowest value

third\_lowest <- sort(unique(values))[3]

cat("Third lowest:", third\_lowest, "\n")

OUTPUT:> # Define the values

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

>

> # Find the second highest value

> second\_highest <- sort(unique(values), decreasing = TRUE)[2]

> cat("Second highest:", second\_highest, "\n")

Second highest: 80

>

> # Find the third lowest value

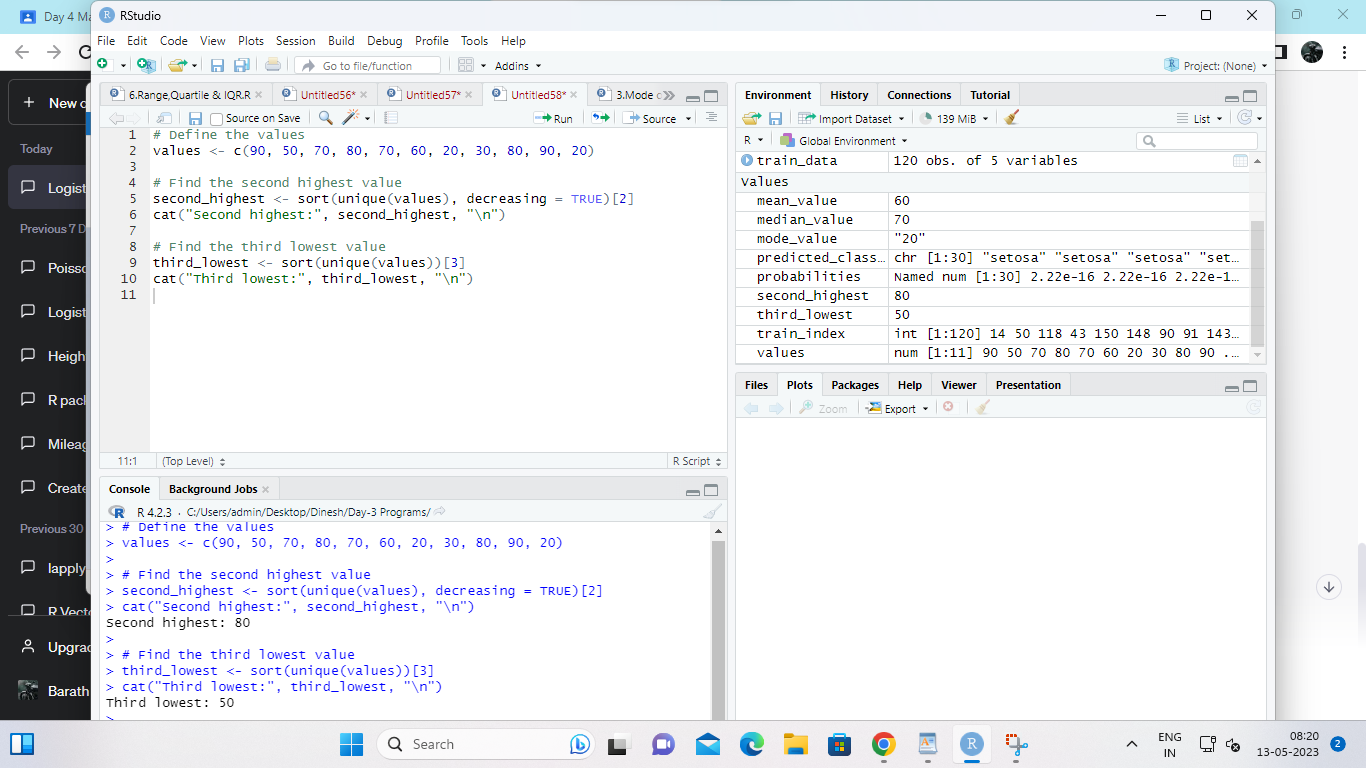
> third\_lowest <- sort(unique(values))[3]

> cat("Third lowest:", third\_lowest, "\n")

Third lowest: 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)**

INPUT:

# Load the airquality dataset

data(airquality)

# Compute the mean temperature

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

# Print the mean temperature

cat("Mean temperature:", mean\_temp, "\n")

OUTPUT:

> # Load the airquality dataset

> data(airquality)

>

> # Compute the mean temperature

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

>

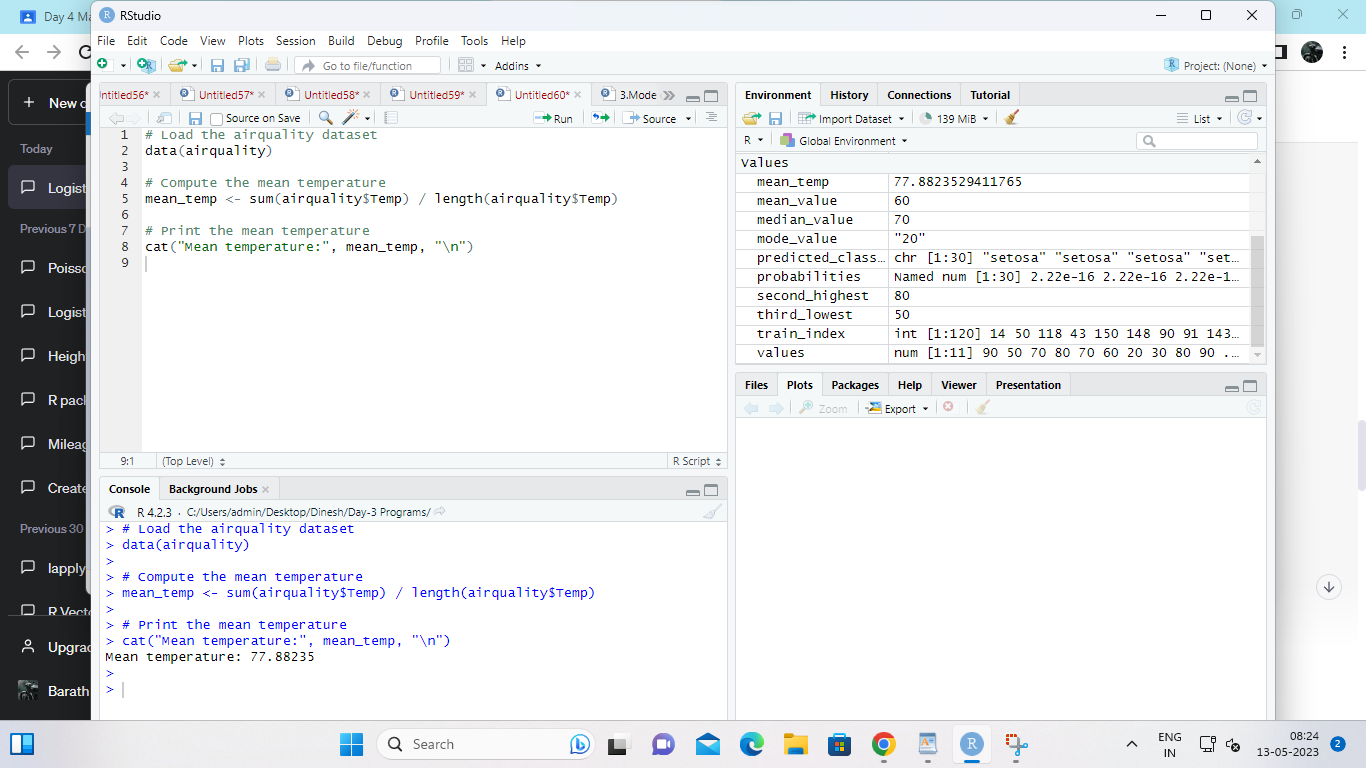
> # Print the mean temperature

> cat("Mean temperature:", mean\_temp, "\n")

Mean temperature: 77.88235

>

>



**ii.Extract the first five rows from airquality.**

INPUT:

# Load the airquality dataset

data(airquality)

# Extract the first five rows

first\_five <- airquality[1:5, ]

# Print the first five rows

print(first\_five)

OUTPUT:

> # Load the airquality dataset

> data(airquality)

>

> # Extract the first five rows

> first\_five <- airquality[1:5, ]

>

> # Print the first five rows

> 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

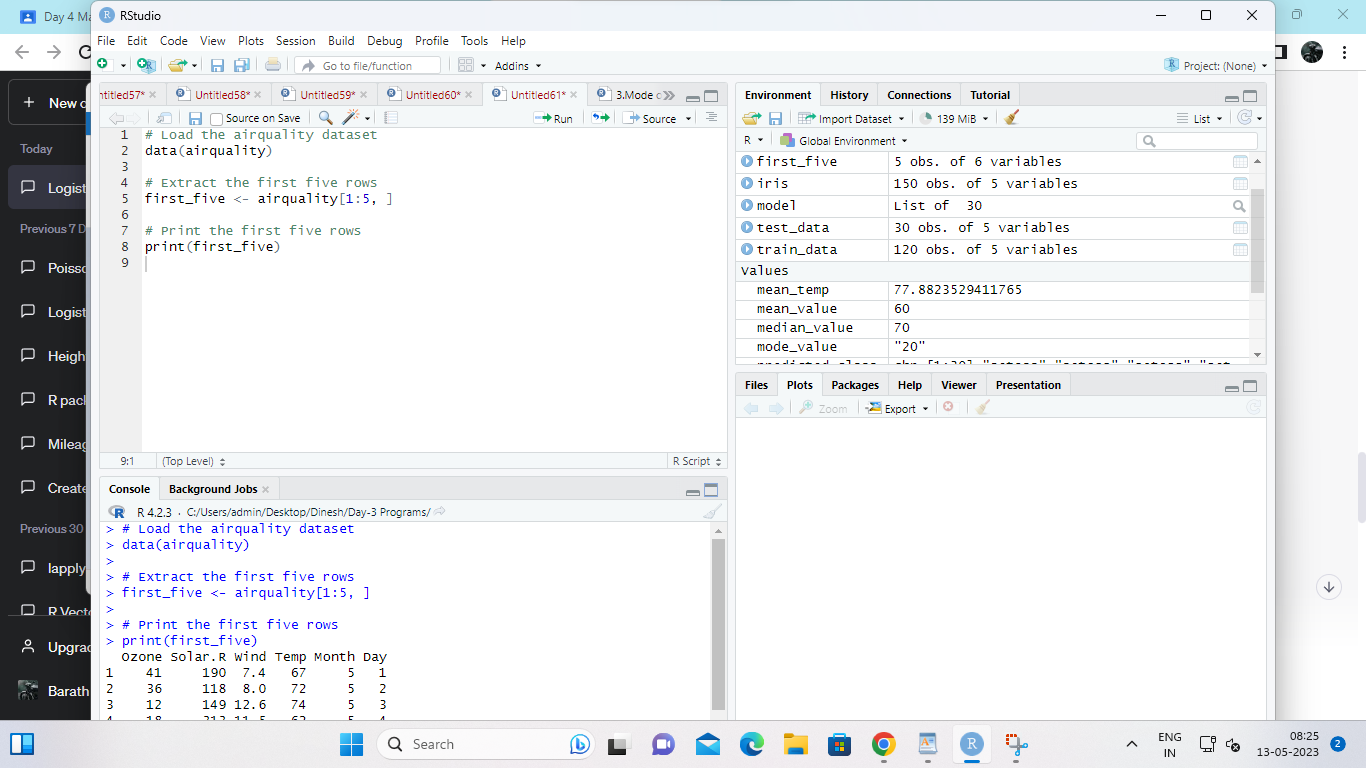
3 12 149 12.6 74 5 3

4 18 313 11.5 62 5 4

5 NA NA 14.3 56 5 5

>

>



iii.Extract all columns from airquality except Temp and Wind

INPUT:

# Load the airquality dataset

data(airquality)

# Extract all columns except Temp and Wind

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

# Print the subset data

print(subset\_data)

OUTPUT:

> # Load the airquality dataset

> data(airquality)

>

> # Extract all columns except Temp and Wind

> subset\_data <- airquality[, !(names(airquality) %in% c("Temp", "Wind"))]

>

> # Print the subset data

> print(subset\_data)

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

6 28 NA 5 6

7 23 299 5 7

8 19 99 5 8

9 8 19 5 9

10 NA 194 5 10

11 7 NA 5 11

12 16 256 5 12

13 11 290 5 13

14 14 274 5 14

15 18 65 5 15

16 14 334 5 16

17 34 307 5 17

18 6 78 5 18

19 30 322 5 19

20 11 44 5 20

21 1 8 5 21

22 11 320 5 22

23 4 25 5 23

24 32 92 5 24

25 NA 66 5 25

26 NA 266 5 26

27 NA NA 5 27

28 23 13 5 28

29 45 252 5 29

30 115 223 5 30

31 37 279 5 31

32 NA 286 6 1

33 NA 287 6 2

34 NA 242 6 3

35 NA 186 6 4

36 NA 220 6 5

37 NA 264 6 6

38 29 127 6 7

39 NA 273 6 8

40 71 291 6 9

41 39 323 6 10

42 NA 259 6 11

43 NA 250 6 12

44 23 148 6 13

45 NA 332 6 14

46 NA 322 6 15

47 21 191 6 16

48 37 284 6 17

49 20 37 6 18

50 12 120 6 19

51 13 137 6 20

52 NA 150 6 21

53 NA 59 6 22

54 NA 91 6 23

55 NA 250 6 24

56 NA 135 6 25

57 NA 127 6 26

58 NA 47 6 27

59 NA 98 6 28

60 NA 31 6 29

61 NA 138 6 30

62 135 269 7 1

63 49 248 7 2

64 32 236 7 3

65 NA 101 7 4

66 64 175 7 5

67 40 314 7 6

68 77 276 7 7

69 97 267 7 8

70 97 272 7 9

71 85 175 7 10

72 NA 139 7 11

73 10 264 7 12

74 27 175 7 13

75 NA 291 7 14

76 7 48 7 15

77 48 260 7 16

78 35 274 7 17

79 61 285 7 18

80 79 187 7 19

81 63 220 7 20

82 16 7 7 21

83 NA 258 7 22

84 NA 295 7 23

85 80 294 7 24

86 108 223 7 25

87 20 81 7 26

88 52 82 7 27

89 82 213 7 28

90 50 275 7 29

91 64 253 7 30

92 59 254 7 31

93 39 83 8 1

94 9 24 8 2

95 16 77 8 3

96 78 NA 8 4

97 35 NA 8 5

98 66 NA 8 6

99 122 255 8 7

100 89 229 8 8

101 110 207 8 9

102 NA 222 8 10

103 NA 137 8 11

104 44 192 8 12

105 28 273 8 13

106 65 157 8 14

107 NA 64 8 15

108 22 71 8 16

109 59 51 8 17

110 23 115 8 18

111 31 244 8 19

112 44 190 8 20

113 21 259 8 21

114 9 36 8 22

115 NA 255 8 23

116 45 212 8 24

117 168 238 8 25

118 73 215 8 26

119 NA 153 8 27

120 76 203 8 28

121 118 225 8 29

122 84 237 8 30

123 85 188 8 31

124 96 167 9 1

125 78 197 9 2

126 73 183 9 3

127 91 189 9 4

128 47 95 9 5

129 32 92 9 6

130 20 252 9 7

131 23 220 9 8

132 21 230 9 9

133 24 259 9 10

134 44 236 9 11

135 21 259 9 12

136 28 238 9 13

137 9 24 9 14

138 13 112 9 15

139 46 237 9 16

140 18 224 9 17

141 13 27 9 18

142 24 238 9 19

143 16 201 9 20

144 13 238 9 21

145 23 14 9 22

146 36 139 9 23

147 7 49 9 24

148 14 20 9 25

149 30 193 9 26

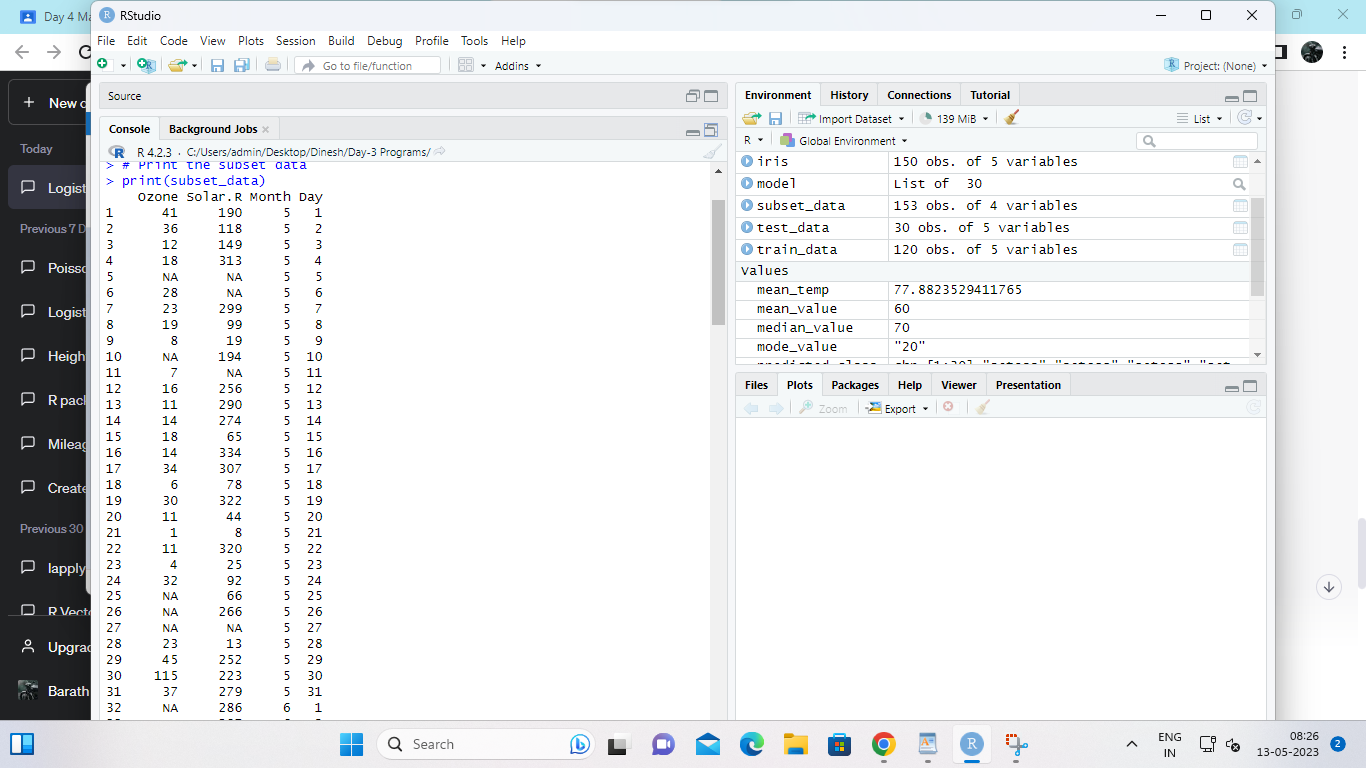
150 NA 145 9 27

151 14 191 9 28

152 18 131 9 29

153 20 223 9 30

>



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

INPUT:

# Load the airquality dataset

data(airquality)

# Find the coldest day

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

# Print the coldest day

cat("Coldest day:", coldest\_day, "\n")

OUTPUT:

> # Load the airquality dataset

> data(airquality)

>

> # Find the coldest day

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

>

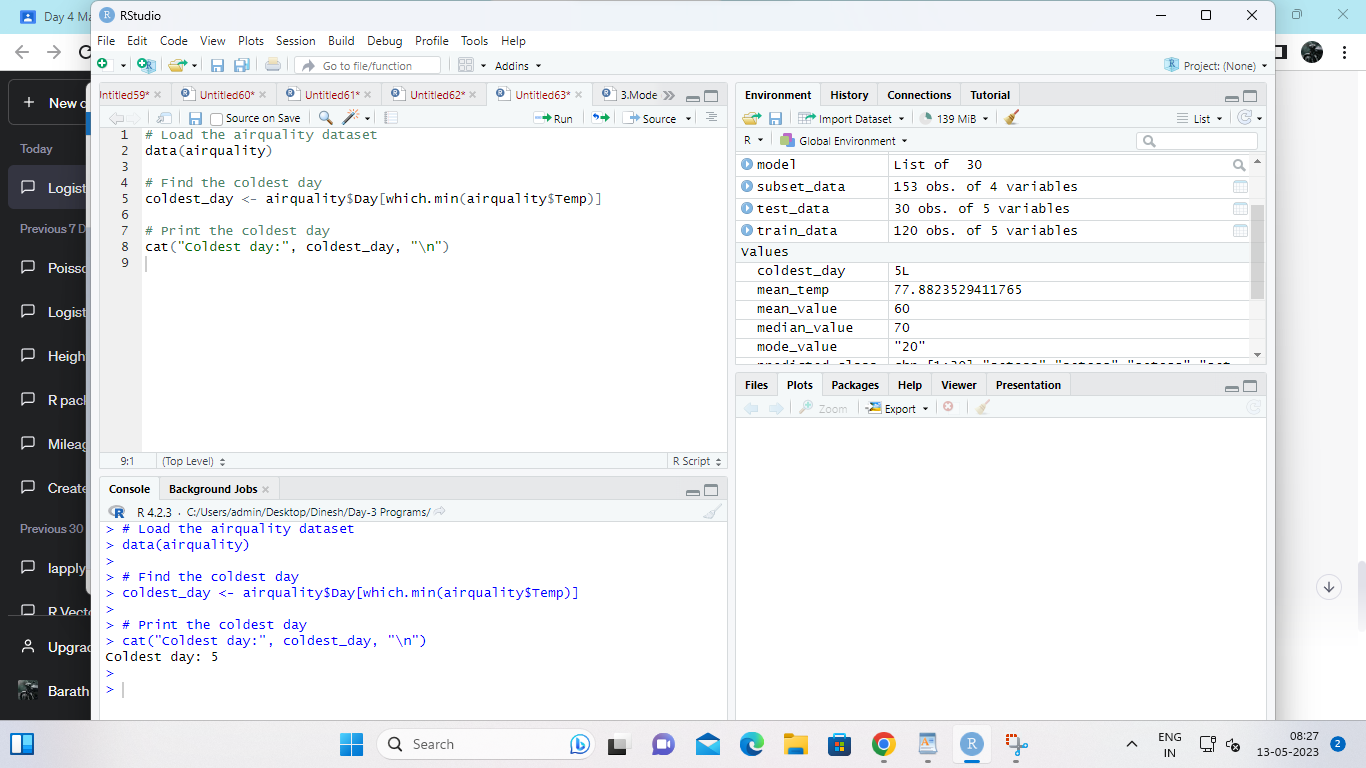
> # Print the coldest day

> cat("Coldest day:", coldest\_day, "\n")

Coldest day: 5

>

>



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

INPUT:

# Load the airquality dataset

data(airquality)

# Count the number of days with wind speed > 17 mph

num\_days\_windy <- sum(airquality$Wind > 17)

# Print the number of days

cat("Number of days with wind speed > 17 mph:", num\_days\_windy, "\n")

OUTPUT:

> # Load the airquality dataset

> data(airquality)

>

> # Find the coldest day

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

>

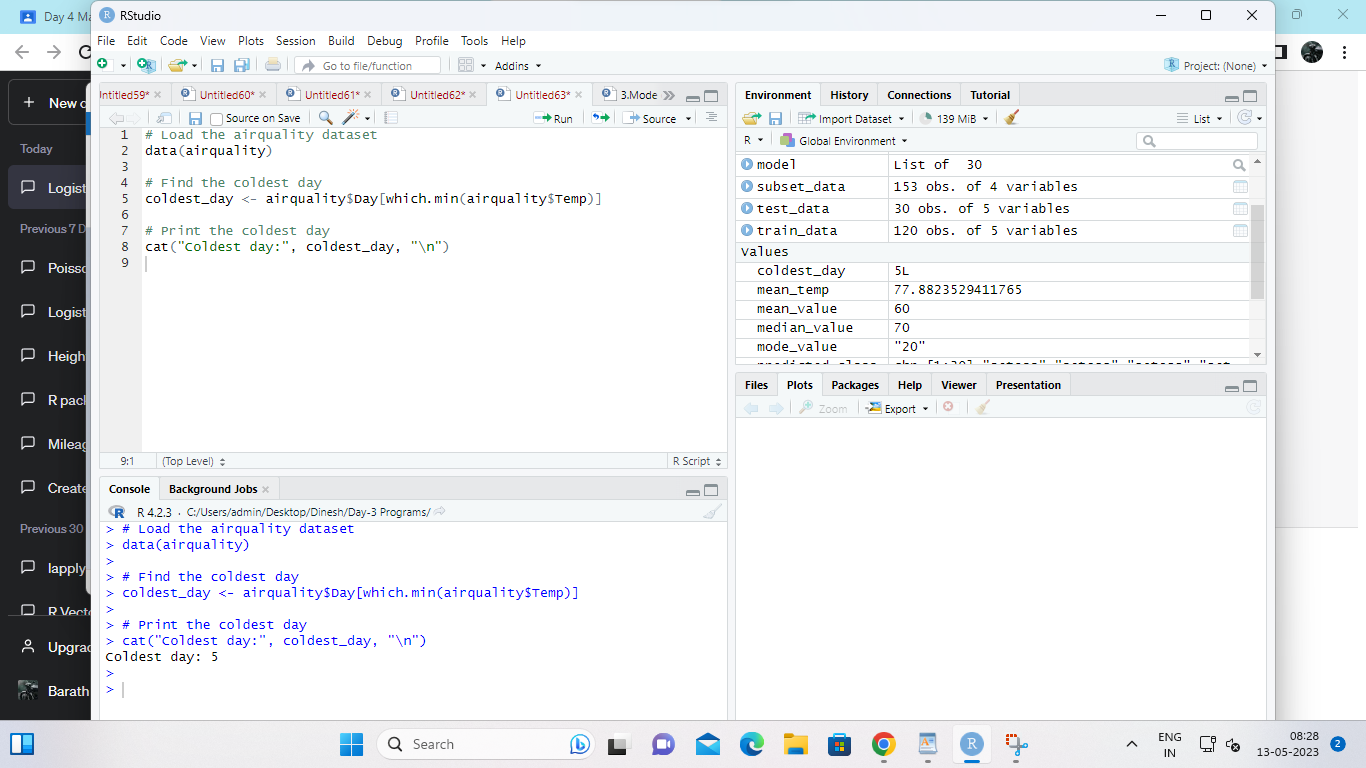
> # Print the coldest day

> cat("Coldest day:", coldest\_day, "\n")

Coldest day: 5

>

>



**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?**

**INPUT;**

data("airquality")

summary(airquality)

library(reshape2)

melted\_data <- melt(airquality)

melted\_data

library(reshape2)

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

melted\_data

library(reshape2)

casted\_data <- cast(melted\_data, Month + Day ~ variable)

casted\_data

library(reshape2)

casted\_data\_mean <- cast(melted\_data, Month ~ variable, mean)

casted\_data\_mean

OUTPUT;

> data("airquality")

> summary(airquality)

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

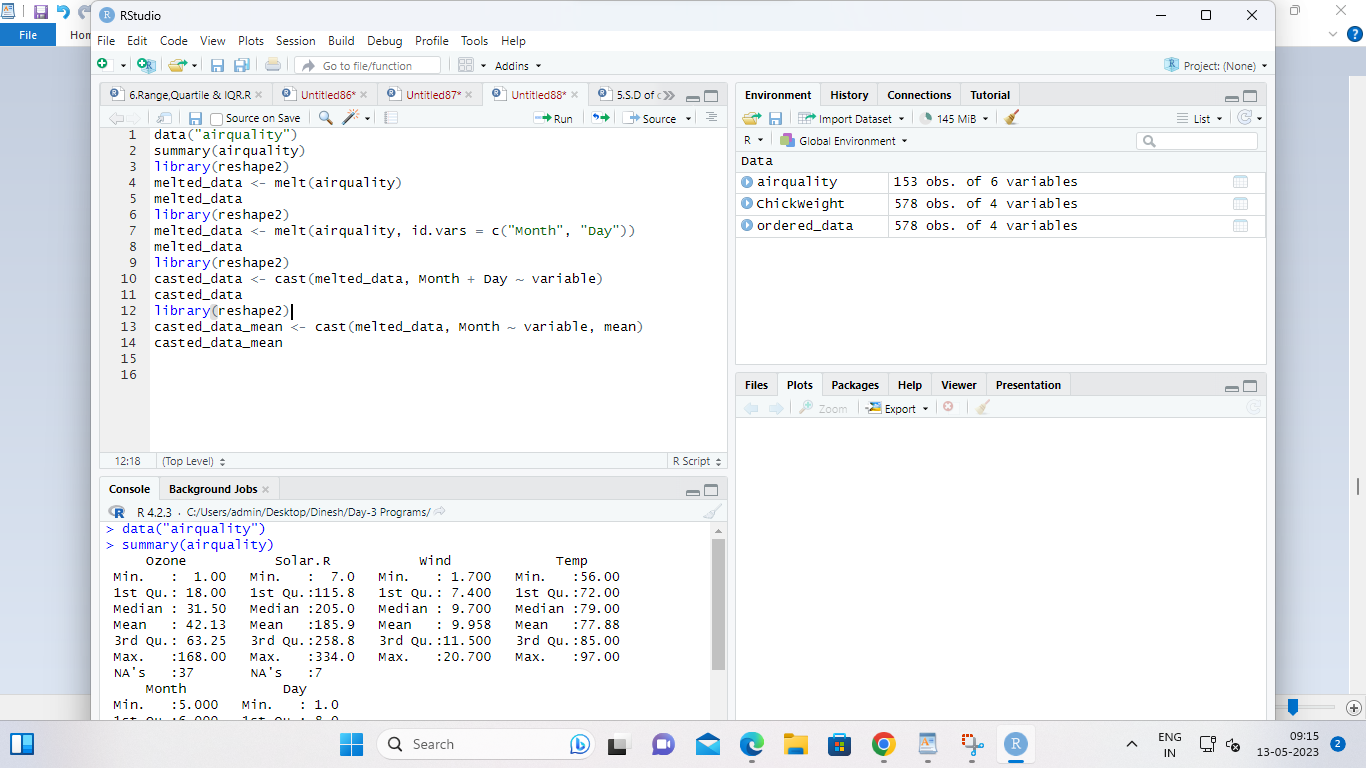
Mean :6.993 Mean :15.8

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

Max. :9.000 Max. :31.0

> library(reshape2)

\



**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**.

INPUT:

# Load airquality dataset

data(airquality)

# Check for missing values in each feature

colSums(is.na(airquality))

# Replace missing values with the mean of that feature if less than 10%, else drop missing values

airquality\_clean <- airquality

for (col in names(airquality\_clean)) {

if (sum(is.na(airquality\_clean[, col])) > 0) {

if (sum(is.na(airquality\_clean[, col])) < 0.1 \* nrow(airquality\_clean)) {

airquality\_clean[is.na(airquality\_clean[, col]), col] <- mean(airquality\_clean[, col], na.rm = TRUE)

} else {

airquality\_clean <- airquality\_clean[!is.na(airquality\_clean[, col]), ]

}

}

}

# Check for missing values after cleaning

colSums(is.na(airquality\_clean))

OUTPUT:

> # Load airquality dataset

> data(airquality)

>

> # Check for missing values in each feature

> colSums(is.na(airquality))

Ozone Solar.R Wind Temp Month Day

37 7 0 0 0 0

>

> # Replace missing values with the mean of that feature if less than 10%, else drop missing values

> airquality\_clean <- airquality

>

> for (col in names(airquality\_clean)) {

+ if (sum(is.na(airquality\_clean[, col])) > 0) {

+ if (sum(is.na(airquality\_clean[, col])) < 0.1 \* nrow(airquality\_clean)) {

+ airquality\_clean[is.na(airquality\_clean[, col]), col] <- mean(airquality\_clean[, col], na.rm = TRUE)

+ } else {

+ airquality\_clean <- airquality\_clean[!is.na(airquality\_clean[, col]), ]

+ }

+ }

+ }

>

> # Check for missing values after cleaning

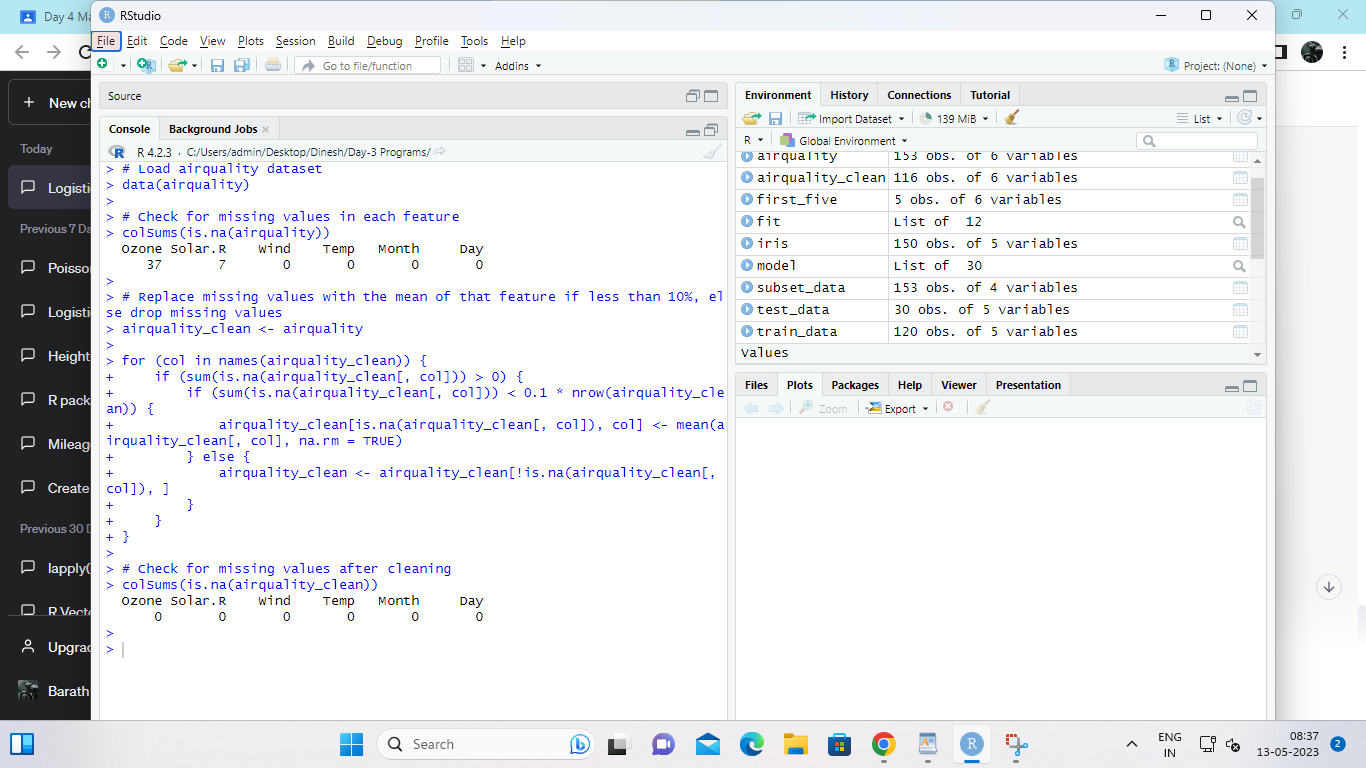
> colSums(is.na(airquality\_clean))

Ozone Solar.R Wind Temp Month Day

0 0 0 0 0 0

>

>



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

**“Solar.R”**

INPUT:

# Fit linear regression model using Least Squares Method on Ozone and Solar.R

fit <- lm(Ozone ~ Solar.R, data = airquality\_clean)

# Print model summary

summary(fit)

OUTPUT:

> # Fit linear regression model using Least Squares Method on Ozone and Solar.R

> fit <- lm(Ozone ~ Solar.R, data = airquality\_clean)

>

> # Print model summary

> summary(fit)

Call:

lm(formula = Ozone ~ Solar.R, data = airquality\_clean)

Residuals:

Min 1Q Median 3Q Max

-48.322 -21.143 -8.033 18.032 119.106

Coefficients:

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

(Intercept) 18.62894 6.67236 2.792 0.006145 \*\*

Solar.R 0.12717 0.03255 3.907 0.000159 \*\*\*

---

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

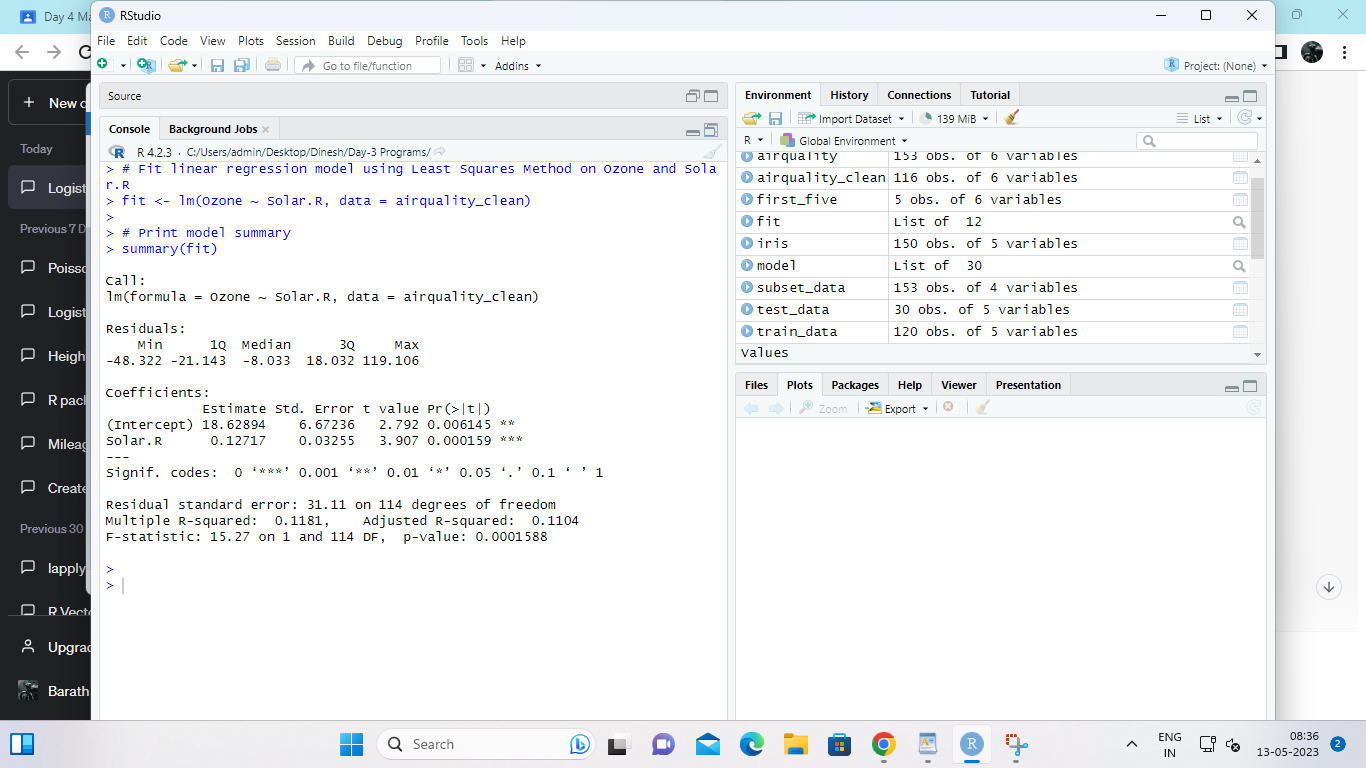
Residual standard error: 31.11 on 114 degrees of freedom

Multiple R-squared: 0.1181, Adjusted R-squared: 0.1104

F-statistic: 15.27 on 1 and 114 DF, p-value: 0.0001588

>

>



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

**above model**

INPUT:# Plot scatter plot between Ozone and Solar.R

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

# Add regression line to the plot

abline(fit, col = "red")

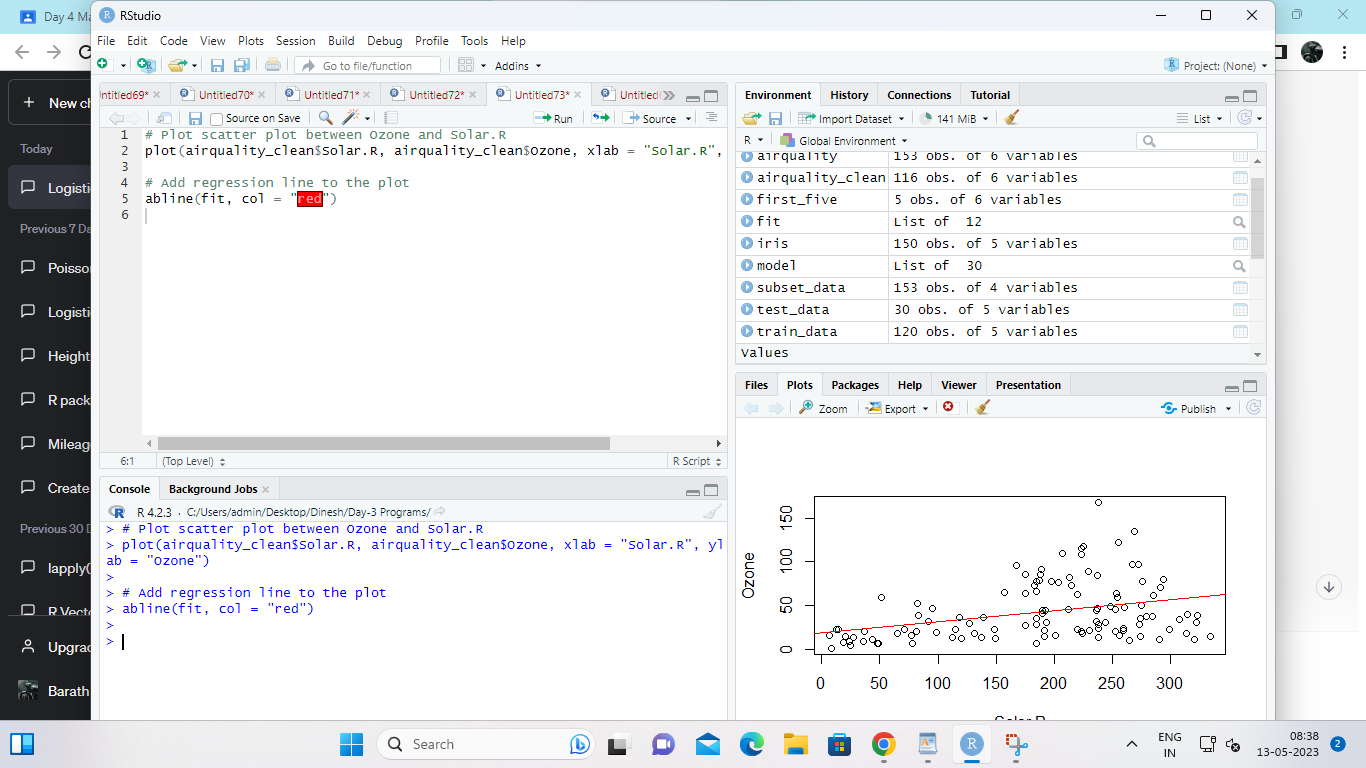
OUTPUT:

# Plot scatter plot between Ozone and Solar.R

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

# Add regression line to the plot

abline(fit, 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&quot;, &quot;Time&quot;, &quot;Diet&quot;   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**

INPUT:

# Loading 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 from order data frame.

ordered\_data <- ChickWeight[order(ChickWeight$diet, ChickWeight$weight),]

last\_6\_records <- tail(ordered\_data, 6)

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

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

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

mean\_weight <- cast(melted\_data, Diet ~ variable, mean)

# (ii)c Perform cast function to display the mode of weight grouped by Diet

library(modeest)

mode\_weight <- cast(melted\_data, Diet ~ variable, mode)

OUTPUT:

library(dplyr)

Attaching package: ‘dplyr’

The following objects are masked from ‘package:stats’:

filter, lag

The following objects are masked from ‘package:base’:

intersect, setdiff, setequal, union

> data("ChickWeight")

> ordered\_data <- arrange(ChickWeight, Diet, weight) # order by Diet and weight

> tail(ordered\_data, 6) # extract the last 6 records

weight Time Chick Diet

573 264 20 50 4

574 264 21 50 4

575 269 20 42 4

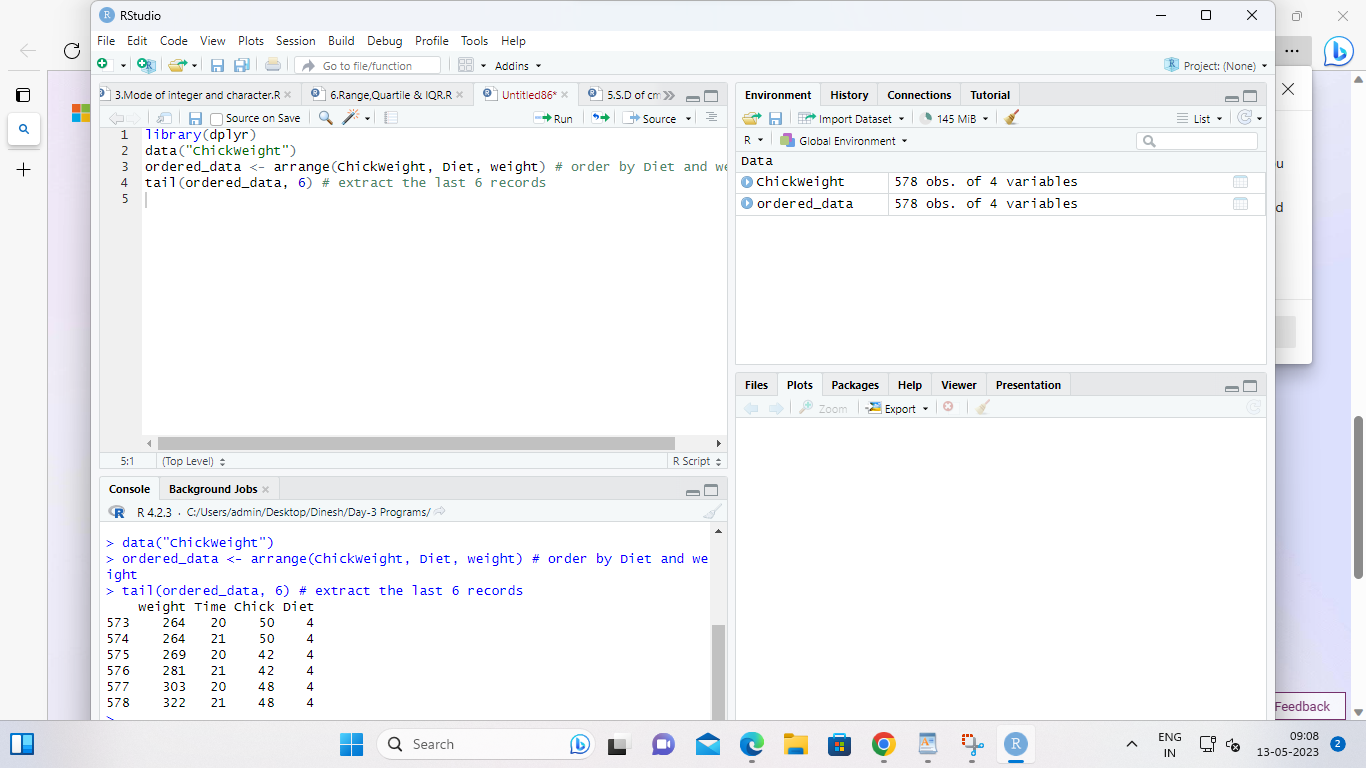
576 281 21 42 4

577 303 20 48 4

578 322 21 48 4

>

>



**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**

INPUT:

# Load the ChickWeight dataset

data(ChickWeight)

# Load the ggplot2 library for plotting

library(ggplot2)

# a. Create Box plot for "weight" grouped by "Diet"

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

geom\_boxplot() +

labs(x = "Diet", y = "Weight") +

ggtitle("Weight by Diet")

# b. Create a Histogram for "weight" features belong to Diet-1 category

ggplot(ChickWeight[ChickWeight$Diet == 1,], aes(x = weight)) +

geom\_histogram(binwidth = 20, color = "black", fill = "lightblue") +

labs(x = "Weight", y = "Count") +

ggtitle("Weight Distribution for Diet 1")

# c. Create Scatter plot for "weight" vs "Time" grouped by Diet

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

geom\_point() +

labs(x = "Time", y = "Weight", color = "Diet") +

ggtitle("Weight vs Time by Diet")

OUTPUT:

