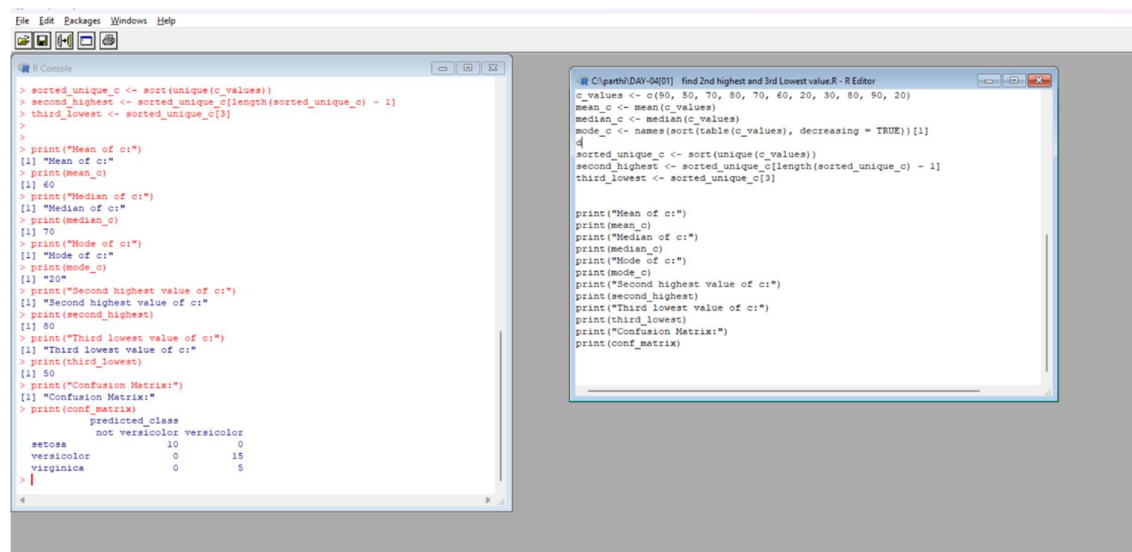


DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
LIST OF EXPERIMENTS
ITA04- STATISTICS WITH R PROGRAMMING
DAY-04

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

(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



The screenshot shows two windows from an R environment. The left window is the R Console, displaying the execution of R code. The right window is the R Editor, showing the source code for a script titled 'find 2nd highest and 3rd Lowest value.R - R Editor'.

R Console Output:

```
> sorted_unique_c <- sort(unique(c_values))
> second_highest <- sorted_unique_c[length(sorted_unique_c) - 1]
> third_lowest <- sorted_unique_c[3]
>
> print("Mean of c:")
[1] "Mean of c:"
> print(mean_c)
[1] 60
> print("Median of c:")
[1] "Median of c:"
> print(median_c)
[1] 70
> print("Mode of c:")
[1] "Mode of c:"
> print(mode_c)
[1] "20"
> print("Second highest value of c:")
[1] "Second highest value of c:"
> print(second_highest)
[1] 80
> print("Third lowest value of c:")
[1] "Third lowest value of c:"
> print(third_lowest)
[1] 50
> print("Confusion Matrix:")
[1] "Confusion Matrix:"
> print(conf_matrix)
      predicted_class
      not versicolor versicolor
setosa             10          0
versicolor         0          15
virginica           0           5
```

R Editor Code:

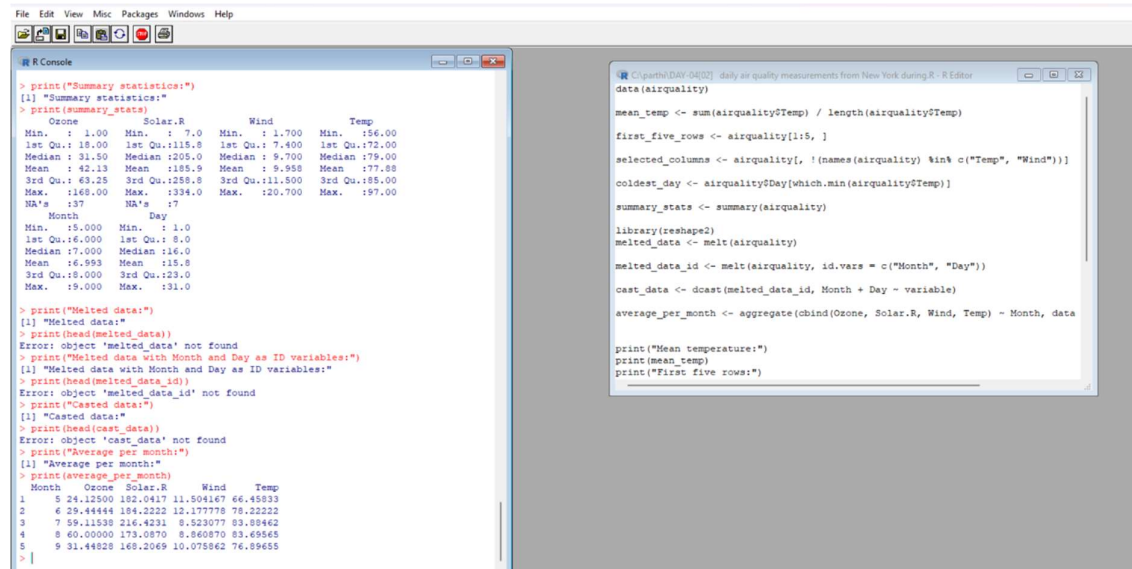
```
## C:\path\DAY-04\01 find 2nd highest and 3rd Lowest value.R - R Editor
c_values <- c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)
mean_c <- mean(c_values)
median_c <- median(c_values)
mode_c <- names(sort(table(c_values), decreasing = TRUE))[1]
4
sorted_unique_c <- sort(unique(c_values))
second_highest <- sorted_unique_c[length(sorted_unique_c) - 1]
third_lowest <- sorted_unique_c[3]

print("Mean of c:")
print(mean_c)
print("Median of c:")
print(median_c)
print("Mode of c:")
print(mode_c)
print("Second highest value of c:")
print(second_highest)
print("Third lowest value of c:")
print(third_lowest)
print("Confusion Matrix:")
print(conf_matrix)
```

2. Explore the air quality 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?
- (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?



```

R Console
> print("Summary statistics:")
[1] "Summary statistics:"
> print(summary_stats)
      Ozone      Solar.R      Wind      Temp
Min.   : 1.00   Min.   : 7.0   Min.   : 1.700   Min.   :56.00
1st Qu.:16.00   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.:1258.8   3rd Qu.:11.500   3rd Qu.:85.00
Max.   :168.00   Max.   :1394.0   Max.   :20.700   Max.   :97.00
NA's   :17      NA's   :7
      Month      Day
Min.   :15.000   Min.   :1.0
1st Qu.:16.000   1st Qu.:8.0
Median :17.000   Median :16.0
Mean   :16.993   Mean   :15.8
3rd Qu.:16.000   3rd Qu.:129.0
Max.   :19.000   Max.   :31.0

> print("Melted data:")
[1] "Melted data:"
> print(head(melted_data))
Error: object 'melted_data' not found
> print("Melted data with Month and Day as ID variables:")
[1] "Melted data with Month and Day as ID variables:"
> print(head(melted_data_id))
Error: object 'melted_data_id' not found
> print("Cast data:")
[1] "Cast data:"
> print(head(cast_data))
Error: object 'cast_data' not found
> print("Average per month:")
[1] "Average per month:"
> print(average_per_month)
      Month      Ozone      Solar.R      Wind      Temp
1      5 24.12500 102.0417 11.504167 66.45833
2      6 29.44444 184.2222 12.177778 78.22222
3      7 59.11538 216.4231  8.523077 83.88462
4      8 60.00000 173.0870  8.860870 83.69565
5      9 31.44828 168.2069 10.075862 76.99655

R Script Editor
C:\path\DAY-04[02] daily air quality measurements from New York during R - R Editor
data(airquality)

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

first_five_rows <- airquality[1:5, ]

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

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

summary_stats <- summary(airquality)

library(reshape2)
melted_data <- melt(airquality)

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

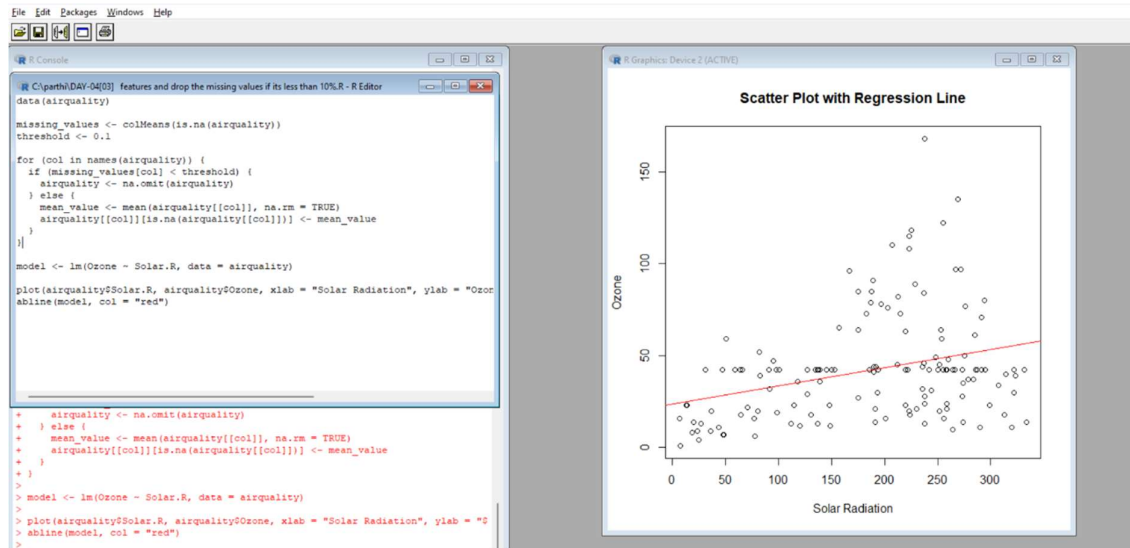
cast_data <- dcast(melted_data_id, Month + Day ~ variable)

average_per_month <- aggregate(cbind(Ozone, Solar.R, Wind, Temp) ~ Month, data

print("Mean temperature:")
print(mean_temp)
print("First five rows:")

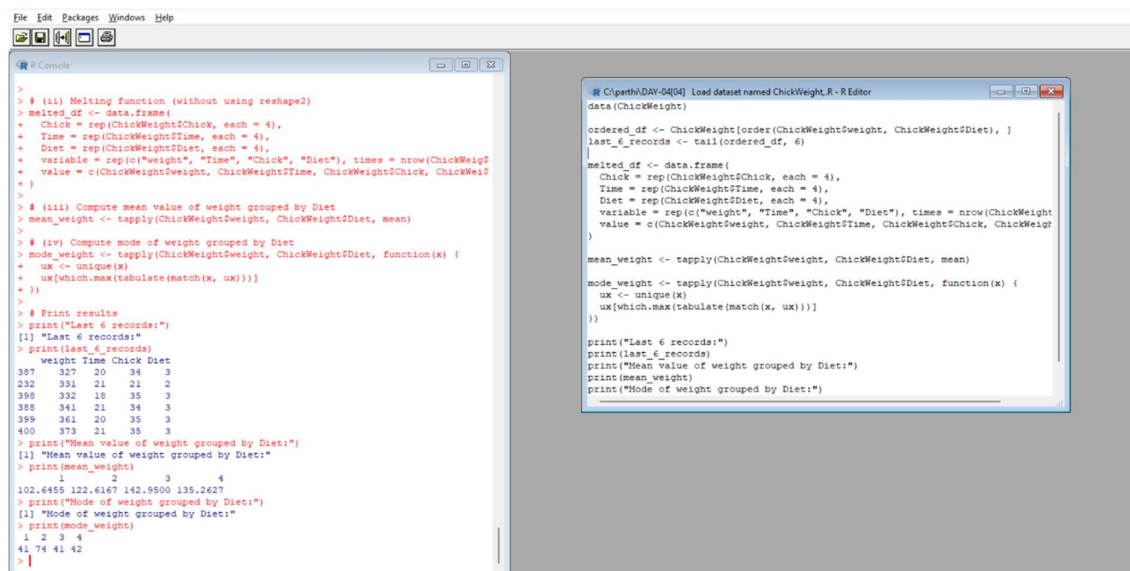
```

- 3.(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

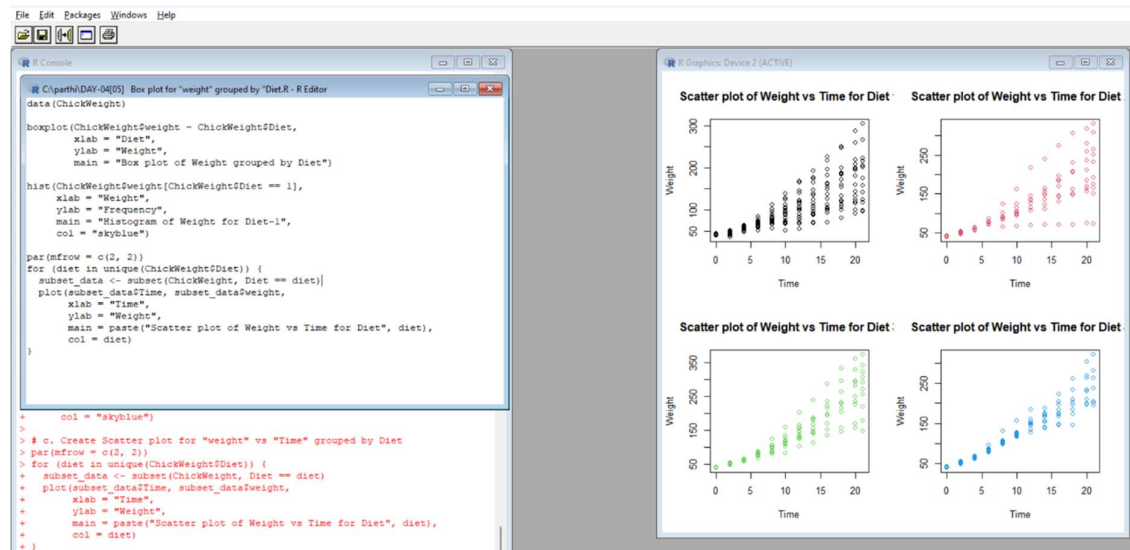


4. Load dataset named ChickWeight,

- 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.
- 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



- 5.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



ANALYTICAL QUESTIONS

- 1.a) Write a R program to sort a given data frame by name and score columns
 - (b) Write R program to find a average score based on “qualify”
- ```

exam_data = data.frame(
 name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura'),
 score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5),
 attempts = c(1, 3, 2, 3, 2, 3, 1, 1),
 qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no'))

```

```

>
> # c. Create Scatter plot for "weight" vs "Time" grouped by Diet
> par(mfrow = c(1, 2))
> for (diet in unique(ChickWeight$Diet)) {
+ subset_data <- subset(ChickWeight, Diet == diet)
+ plot(subset_data$Time, subset_data$weight,
+ xlab = "Time",
+ ylab = "Weight",
+ main = paste("Scatter plot of Weight vs Time for Diet", diet),
+ col = diet)
+ }
> exam_data <- data.frame(
+ name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura'),
+ score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5),
+ attempts = c(1, 3, 2, 3, 2, 3, 1, 1),
+ qualify = c('yes', 'no', 'yes', 'no', 'yes', 'yes', 'no')
+)
> sorted_df <- exam_data[order(exam_data$name, exam_data$score),]
>
> average_score <- tapply(exam_data$score, exam_data$qualify, mean)
>
> print("Sorted data frame by name and score:")
[1] "Sorted data frame by name and score:"
> print(sorted_df)
 name score attempts qualify
1 Anastasia 12.5 1 yes
2 Dima 9.0 3 no
3 Emily 9.0 2 no
4 James 12.0 3 no
5 Katherine 16.5 2 yes
6 Laura 13.5 1 no
7 Matthew 14.5 1 yes
8 Michael 20.0 3 yes
>
> print("Average score based on qualify:")
[1] "Average score based on qualify:"
> print(average_score)
no yes
10.875 15.875
>

```

2. Write a R program to create inner, outer, left, right join(merge) from given two data frames  
df1 = data. frame (numid = c(12, 14, 10, 11))  
df2 = data. frame (numid = c(13, 15, 11, 12))

```

> left_join <- merge(df1, df2, by = "numid", all.x = TRUE)
>
> right_join <- merge(df1, df2, by = "numid", all.y = TRUE)
>
> print("Inner Join:")
[1] "Inner Join:"
> print(inner_join)
 numid
1 11
2 12
>
> print("Outer Join:")
[1] "Outer Join:"
> print(outer_join)
 numid
1 10
2 11
3 12
4 13
5 14
6 15
>
> print("Left Join:")
[1] "Left Join:"
> print(left_join)
 numid
1 10
2 11
3 12
4 14
>
> print("Right Join:")
[1] "Right Join:"
> print(right_join)
 numid
1 11
2 12
3 13
4 15
>

```

3.(a) Write a R program to replace NA values with 3 in a given data frame and sort a given data frame by score columns.

(b) Write R program to find a average score who qualified in first attempts

```
exam_data = data.frame(
 name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura'),
 score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5),
 attempts = c(1, NA, 2, NA, 2, NA, 1, 1),
 qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no'))
```

The screenshot shows the R Studio interface with two windows. The 'R Console' window on the left displays the execution of the R script, showing the creation of the 'exam\_data' data frame, the replacement of NA values with 3, the sorting of the data frame by score, and the calculation of the average score of those who qualified in the first attempt. The 'R Editor' window on the right shows the source code for the script.

```
> print("Right Join:")
[1] "Right Join:"
> print(right_join)
 numid
1 11
2 12
3 13
4 15
> exam_data <- data.frame(
+ name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Ma2
+ score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5),
+ attempts = c(1, NA, 2, NA, 2, NA, 1, 1),
+ qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no')
+)
>
> exam_data$attempts[is.na(exam_data$attempts)] <- 3
>
> sorted_data <- exam_data[order(exam_data$score),]
>
> print("Sorted Data Frame:")
[1] "Sorted Data Frame:"
> print(sorted_data)
 name score attempts qualify
2 Dima 9.0 3 no
5 Emily 9.0 2 no
4 James 12.0 3 no
1 Anastasia 12.5 1 yes
8 Laura 13.5 1 no
7 Matthew 14.5 1 yes
3 Katherine 16.5 2 yes
6 Michael 20.0 3 yes
>
> qualified_first_attempt <- subset(exam_data, qualify == "yes" & attempts == 1)
> average_score <- mean(qualified_first_attempt$score, na.rm = TRUE)
>
> print("Average Score of Those Who Qualified in First Attempt:")
[1] "Average Score of Those Who Qualified in First Attempt:"
> print(average_score)
[1] 13.5
..
```