B. Maheshwar(192411105)

CSA1618– Data Warehouse and Data Mining
Day 1 Lab Programs (R)

Program 1: Approximate Median for Grouped Data

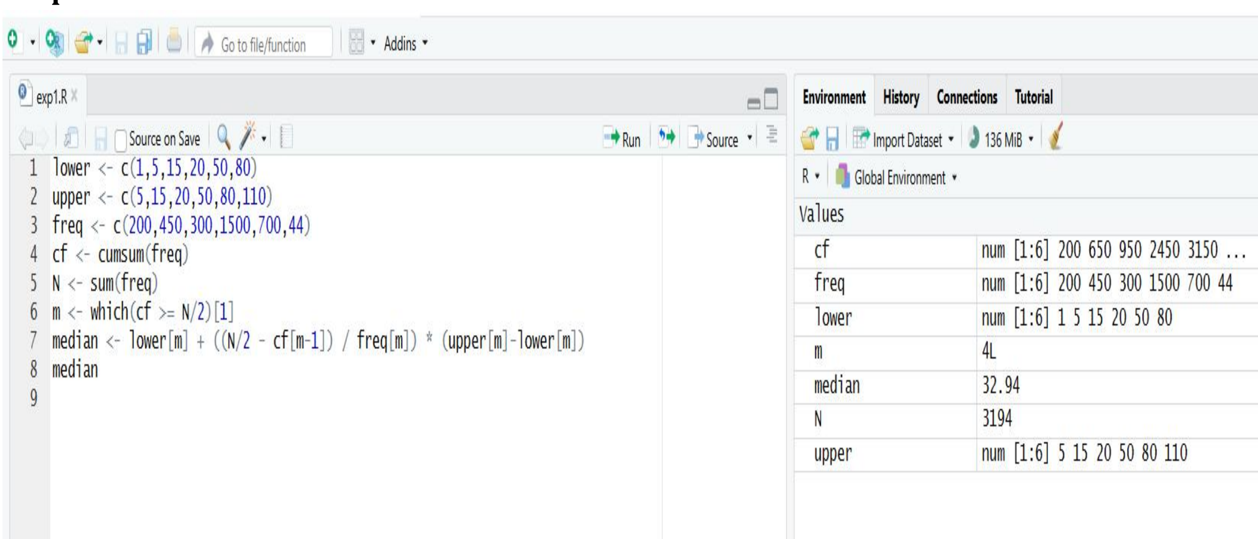
Aim:
To compute the approximate median for grouped frequency data.

Algorithm:
1. Store class intervals and frequencies.
2. Find cumulative frequency.
3. Identify median class.
4. Apply median formula.

R Program:

```
lower <- c(1,5,15,20,50,80)
upper <- c(5,15,20,50,80,110)
freq <- c(200,450,300,1500,700,44)
cf <- cumsum(freq)
N <- sum(freq)
m <- which(cf >= N/2)[1]
median <- lower[m] + ((N/2 - cf[m-1]) / freq[m]) * (upper[m]-lower[m])
median
```

Output:



Program 2: Mean, Median, Mode, Quartiles

Aim:

To compute mean, median, mode, midrange and quartiles.

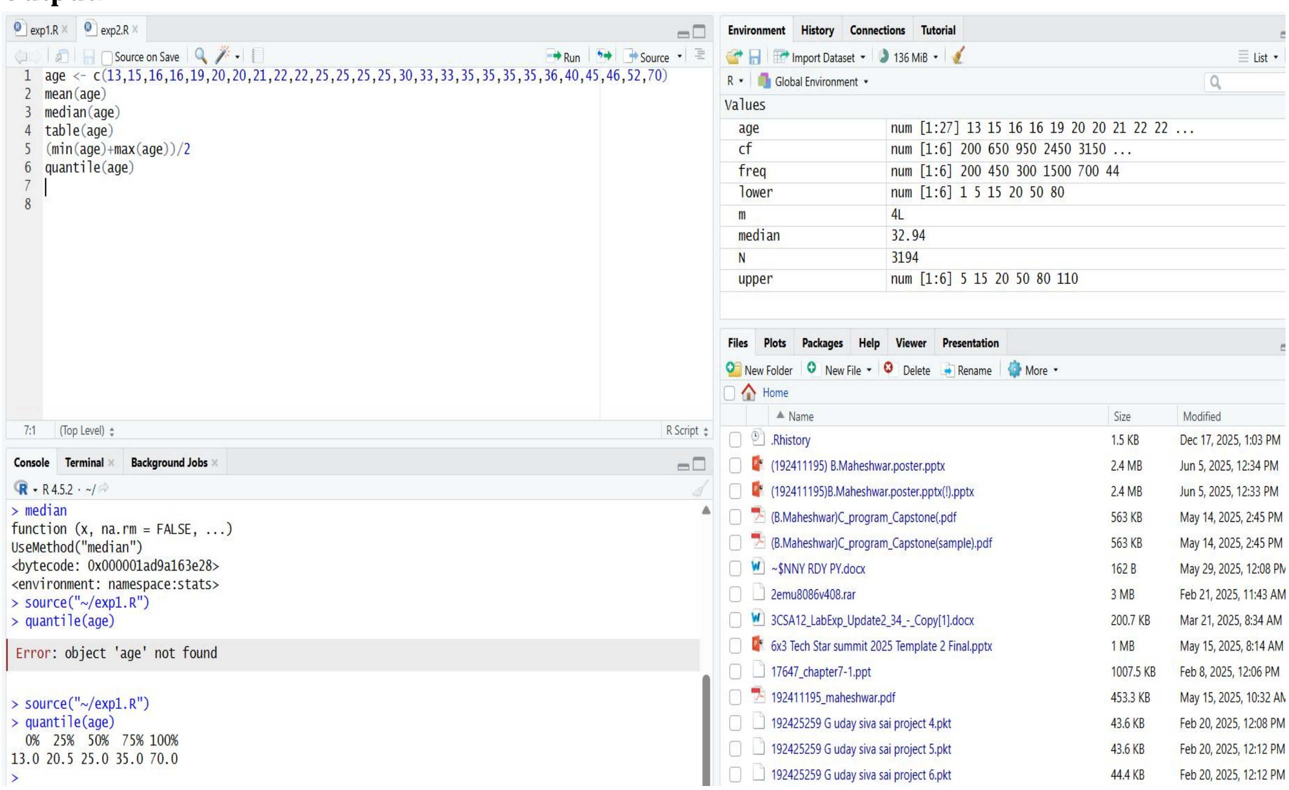
Algorithm:

1. Read sorted age data.
2. Compute mean and median.
3. Find mode using frequency count.
4. Calculate midrange, Q1, and Q3.

R Program:

```
age <- c(13,15,16,16,19,20,20,21,22,22,25,25,25,25,30,33,33,35,35,35,35,36,40,45,46,52,70)
mean(age)
median(age)
table(age)
(min(age)+max(age))/2
quantile(age)
```

Output:



Program 3: Normalization

Aim:

To normalize data using Min-Max and Z-score.

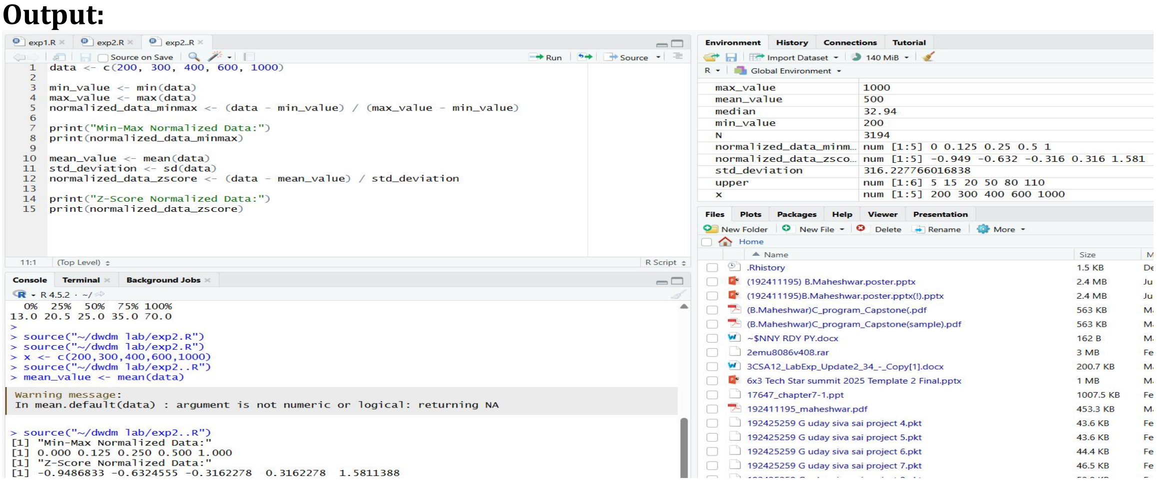
Algorithm:

- Read numerical data.
- Apply Min–Max normalization.
- Compute mean and standard deviation.
- Apply Z-score normalization.

R Program:

```
data <- c(200, 300, 400, 600, 1000)

min_value <- min(data)

max_value <- max(data)

normalized_data_minmax <- (data - min_value) / (max_value - min_value)

print("Min-Max Normalized Data:")

print(normalized_data_minmax)

mean_value <- mean(data)

std_deviation <- sd(data)

normalized_data_zscore <- (data - mean_value) / std_deviation

print("Z-Score Normalized Data:")

print(normalized_data_zscore)
```

Output:



Program 4: Data Smoothing

Aim:

To smooth data using bin mean, median, boundaries.

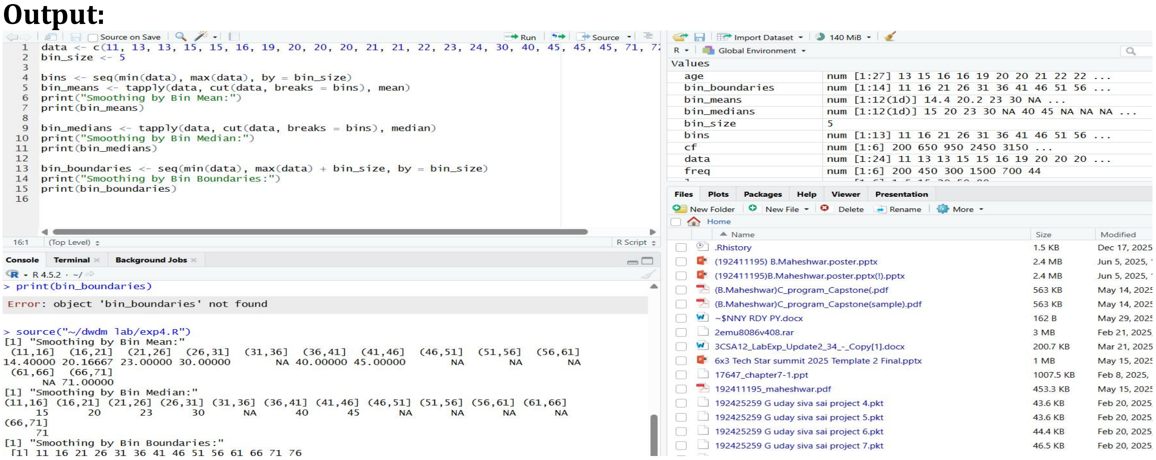
Algorithm:

1. Read sorted data values.
2. Divide data into bins.
3. Compute bin mean and bin median.
4. Determine bin boundaries.

R Program:

```
data <- c(11, 13, 13, 15, 15, 16, 19, 20, 20, 20, 21, 21, 22, 23, 24, 30, 40, 45, 45, 45, 71, 72, 73, 75)

bin_size <- 5

bins <- seq(min(data), max(data), by = bin_size)

bin_means <- tapply(data, cut(data, breaks = bins), mean)

print("Smoothing by Bin Mean:")

print(bin_means)

bin_medians <- tapply(data, cut(data, breaks = bins), median)

print("Smoothing by Bin Median:")

print(bin_medians)

bin_boundaries <- seq(min(data), max(data) + bin_size, by = bin_size)

print("Smoothing by Bin Boundaries:")

print(bin_boundaries)
```

Output:



Program 5: Descriptive Statistics & Plots

Aim:
To compute statistics and draw plots.

Algorithm:

- Read age and %fat data.
- Compute mean, median, and standard deviation.
- Draw boxplots.
- Plot scatter and Q–Q plots.

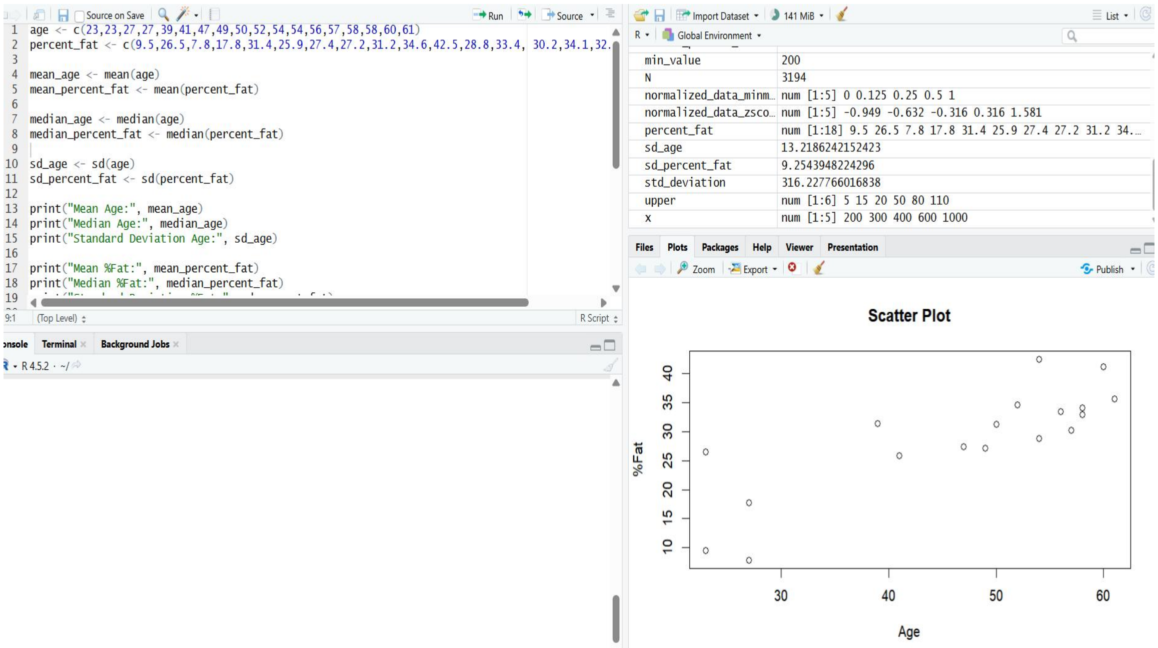
R Program:

```
age <- c(23,23,27,27,39,41,47,49,50,52,54,54,56,57,58,58,60,61)

percent_fat <- c(9.5,26.5,7.8,17.8,31.4,25.9,27.4,27.2,31.2,34.6,42.5,28.8,33.4,
30.2,34.1,32.9,41.2,35.7)

mean_age <- mean(age)

mean_percent_fat <- mean(percent_fat)

median_age <- median(age)

median_percent_fat <- median(percent_fat)

sd_age <- sd(age)

sd_percent_fat <- sd(percent_fat)

print("Mean Age:", mean_age)

print("Median Age:", median_age)

print("Standard Deviation Age:", sd_age)

print("Mean %Fat:", mean_percent_fat)

print("Median %Fat:", median_percent_fat)

print("Standard Deviation %Fat:", sd_percent_fat)

boxplot(age, main="Boxplot of Age")

boxplot(percent_fat, main="Boxplot of %Fat")

plot(age, percent_fat, main="Scatter Plot", xlab="Age", ylab="%Fat")

qqnorm(age)

qqline(age, col = 2)
```

qqnorm(percent_fat)

qqline(percent_fat, col = 2)

Output:



Program 6: Normalization Techniques

Aim:

To perform normalization on age value.

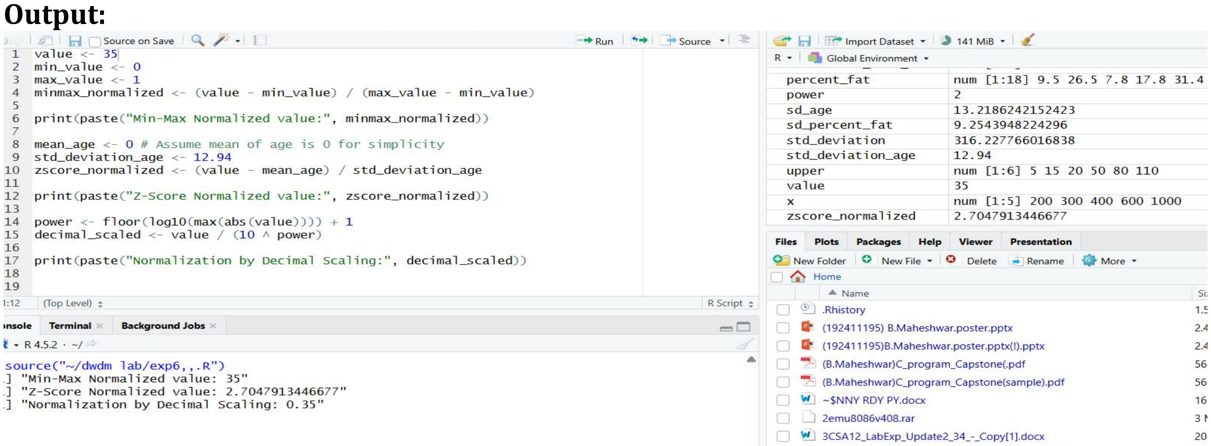
Algorithm:

1. Read the given value.
2. Apply Min–Max normalization.
3. Apply Z-score normalization.
4. Apply decimal scaling.

R Program:

```
value <- 35

min_value <- 0

max_value <- 1

minmax_normalized <- (value - min_value) / (max_value - min_value)

print(paste("Min-Max Normalized value:", minmax_normalized))

mean_age <- 0 # Assume mean of age is 0 for simplicity

std_deviation_age <- 12.94

zscore_normalized <- (value - mean_age) / std_deviation_age

print(paste("Z-Score Normalized value:", zscore_normalized))

power <- floor(log10(max(abs(value)))) + 1

decimal_scaled <- value / (10 ^ power)

print(paste("Normalization by Decimal Scaling:", decimal_scaled))
```

Output:



Program 7: Mean Median Mode

Aim:

To compute mean, median and mode.

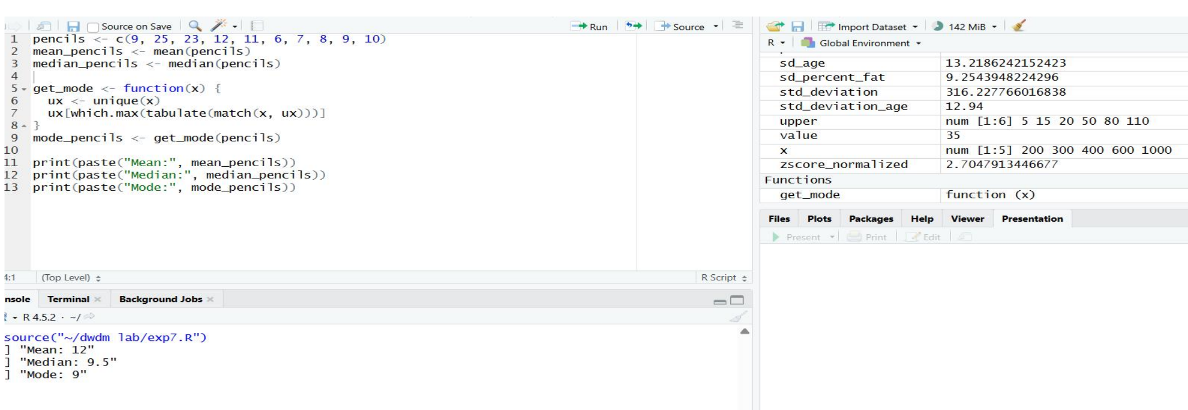
Algorithm:

- Store pencil counts in a vector.
- Compute mean and median.
- Identify mode.
- Display results.

R Program:

```
pencils <- c(9, 25, 23, 12, 11, 6, 7, 8, 9, 10)

mean_pencils <- mean(pencils)

median_pencils <- median(pencils)

get_mode <- function(x) {

 ux <- unique(x)

 ux[which.max(tabulate(match(x, ux)))]

}

mode_pencils <- get_mode(pencils)

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

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

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

Output:



Program 8: Scatter Plot

Aim:

To draw scatter plot for given data.

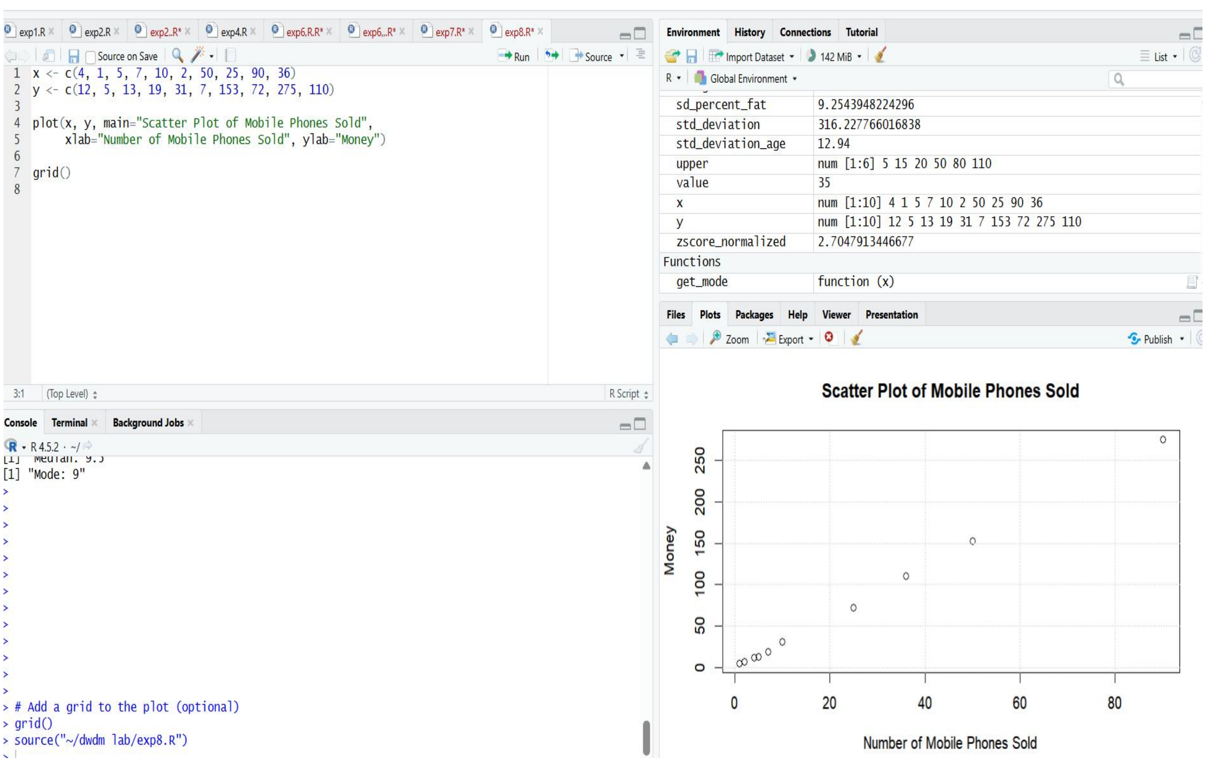
Algorithm:

1. Read x and y values.
2. Plot x versus y.
3. Label axes and display graph.

R Program:

```
x <- c(4, 1, 5, 7, 10, 2, 50, 25, 90, 36)

y <- c(12, 5, 13, 19, 31, 7, 153, 72, 275, 110)

plot(x, y, main="Scatter Plot of Mobile Phones Sold",

   xlab="Number of Mobile Phones Sold", ylab="Money")

grid()
```

Output:



Program 9: Data Partitioning

Aim:

To partition data and draw histogram.

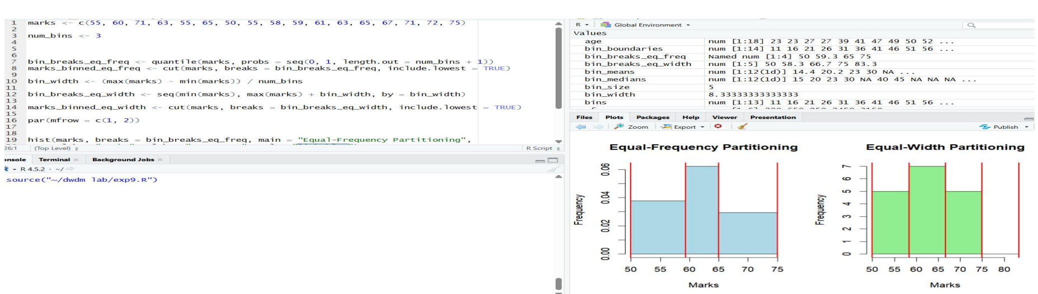
Algorithm:

- Read student marks.
- Divide data into equal-frequency bins.
- Divide data into equal-width bins.
- Plot histograms.

R Program:

```
marks <- c(55, 60, 71, 63, 55, 65, 50, 55, 58, 59, 61, 63, 65, 67, 71, 72, 75)

num_bins <- 3

bin_breaks_eq_freq <- quantile(marks, probs = seq(0, 1, length.out = num_bins + 1))

marks_binned_eq_freq <- cut(marks, breaks = bin_breaks_eq_freq, include.lowest = TRUE)

bin_width <- (max(marks) - min(marks)) / num_bins

bin_breaks_eq_width <- seq(min(marks), max(marks) + bin_width, by = bin_width)

marks_binned_eq_width <- cut(marks, breaks = bin_breaks_eq_width, include.lowest =
TRUE)

par(mfrow = c(1, 2))

hist(marks, breaks = bin_breaks_eq_freq, main = "Equal-Frequency Partitioning",

   xlab = "Marks", ylab = "Frequency", col = "lightblue")

abline(v = bin_breaks_eq_freq, col = "red", lwd = 2)

hist(marks, breaks = bin_breaks_eq_width, main = "Equal-Width Partitioning",

   xlab = "Marks", ylab = "Frequency", col = "lightgreen")

abline(v = bin_breaks_eq_width, col = "red", lwd = 2)\
```

Output:



Program 10: IQR and Standard Deviation

Aim:

To compute IQR and SD.

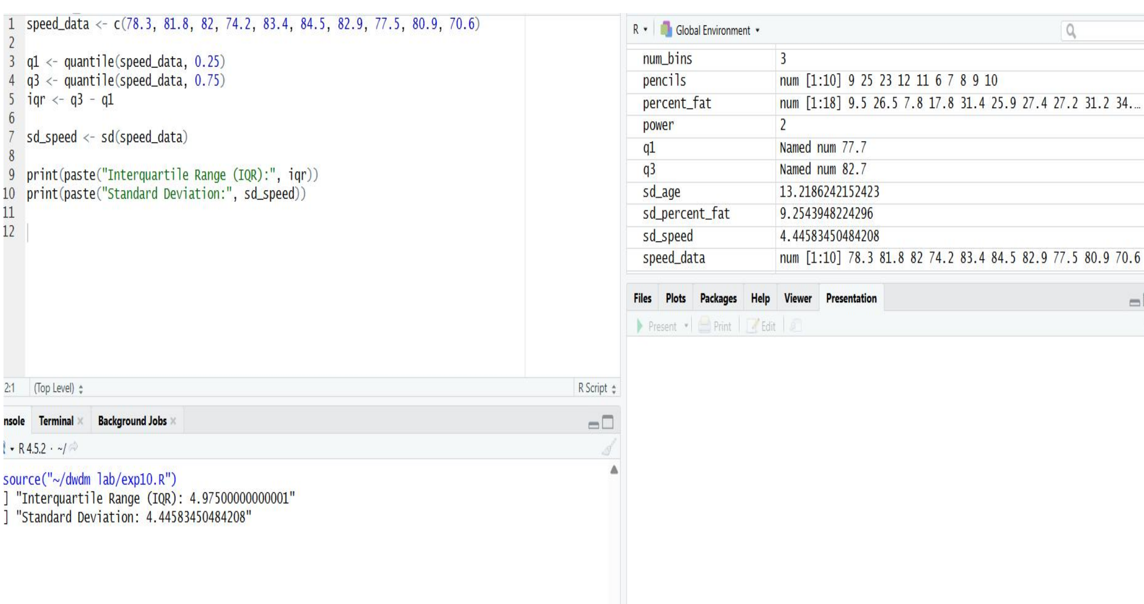
Algorithm:

1. Read speed data.
2. Calculate Q1 and Q3.
3. Compute IQR.
4. Calculate standard deviation.

R Program:

```
speed_data <- c(78.3, 81.8, 82, 74.2, 83.4, 84.5, 82.9, 77.5, 80.9, 70.6)

q1 <- quantile(speed_data, 0.25)

q3 <- quantile(speed_data, 0.75)

iqr <- q3 - q1

sd_speed <- sd(speed_data)

print(paste("Interquartile Range (IQR):", iqr))

print(paste("Standard Deviation:", sd_speed))
```

Outout:



Program 11: Quartiles

Aim:

To find Q1 and Q3.

Algorithm:

- Read sorted age data.
- Find quartile positions.
- Identify Q1 and Q3 values.
- Display results.

R Program:

```
age_data <- c(13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70)

total_data_points <- length(age_data)

q1_position <- (total_data_points + 1) / 4

q3_position <- 3 * q1_position

q1 <- age_data[ceiling(q1_position)]

q3 <- age_data[ceiling(q3_position)]

print(paste("Approximate First Quartile (Q1):", q1))

print(paste("Approximate Third Quartile (Q3):", q3))
```

Output: