

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:

The screenshot shows the RStudio interface with the following details:

- Code Editor:** The left pane displays the R script named "exp1.R" containing the provided R code.
- Environment Tab:** The right pane shows the R environment with the following variables and their values:

Variable	Value
cf	[1:6] 200 650 950 2450 3150 ...
freq	[1:6] 200 450 300 1500 700 44
lower	[1:6] 1 5 15 20 50 80
m	4L
median	32.94
N	3194
upper	[1:6] 5 15 20 50 80 110

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

Output:

The screenshot shows the RStudio interface with the following components:

- R Script pane:** Displays the R code provided above.
- Console pane:** Shows the execution of the R code. It starts with the code, then displays an error message: "Error: object 'age' not found". After this, it runs the code again successfully, showing the output of `quantile(age)`.
- Global Environment pane:** Shows the following objects and their values:

Object	Type	Value
age	num [1:27]	13 15 16 16 19 20 20 21 22 22 ...
cf	num [1:6]	200 650 950 2450 3150 ...
freq	num [1:6]	200 450 300 1500 700 44
lower	num [1:6]	1 5 15 20 50 80
m	4L	32.94
median	32.94	32.94
N	3194	3194
upper	num [1:6]	5 15 20 50 80 110

- File Explorer pane:** Shows a list of files in the current directory, including various presentation files (pptx, pdf), a rar archive, and word documents.

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:

The screenshot shows the RStudio interface with the R Script pane containing the R code for normalization, and the Environment pane displaying the resulting variables and their values.

R Script:

```
1 data <- c(200, 300, 400, 600, 1000)
2 min_value <- min(data)
3 max_value <- max(data)
4 normalized_data_minmax <- (data - min_value) / (max_value - min_value)
5 print("Min-Max Normalized Data:")
6 print(normalized_data_minmax)
7
8 mean_value <- mean(data)
9 std_deviation <- sd(data)
10 normalized_data_zscore <- (data - mean_value) / std_deviation
11 print("Z-Score Normalized Data:")
12 print(normalized_data_zscore)
```

Environment:

Name	Value
max_value	1000
mean_value	500
median	32.94
min_value	200
N	3194
normalized_data_minmax	num [1:5] 0 0.125 0.25 0.5 1
normalized_data_zscore	num [1:5] -0.949 -0.632 -0.316 0.316 1.581
std_deviation	316.2277660138838
upper	num [1:5] 15 20 50 80 110
x	num [1:5] 200 300 400 600 1000

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, 13, 15, 15, 16, 19, 20, 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:

```
1 data <- c(11, 13, 13, 13, 15, 15, 16, 19, 20, 20, 20, 20, 21, 21, 22, 23, 24, 30, 40, 45, 45, 45, 71, 72, 73, 75)
2 bin_size <- 5
3
4 bins <- seq(min(data), max(data), by = bin_size)
5 bin_means <- tapply(data, cut(data, breaks = bins), mean)
6 print("Smoothing by Bin Mean:")
7 print(bin_means)
8
9 bin_medians <- tapply(data, cut(data, breaks = bins), median)
10 print("Smoothing by Bin Median:")
11 print(bin_medians)
12
13 bin_boundaries <- seq(min(data), max(data) + bin_size, by = bin_size)
14 print("Smoothing by Bin Boundaries:")
15 print(bin_boundaries)
16
```

R Script

Values

data	num [1:27]	13 15 16 16 19 20 20 21 22 22 ...
bin_boundaries	num [1:14]	11 16 21 26 31 36 41 46 51 56 ...
bin_means	num [1:12 (1d)]	14.4 20.2 23 30 NA ...
bin_medians	num [1:12 (1d)]	15 20 23 30 NA 40 45 NA NA ...
bin_size	5	
cf	num [1:13]	11 16 21 26 31 36 41 46 51 56 ...
data	num [1:24]	11 13 13 15 16 19 20 20 20 ...
freq	num [1:6]	200 350 300 1500 700 44

Files

Name	Size	Modified
Rhistory	1.5 KB	Dec 17, 2025
(192411195) B.Maheshwar.poster.pptx	2.4 MB	Jun 5, 2025, ·
(192411195) B.Maheshwar.poster.pptx (0).pptx	2.4 MB	Jun 5, 2025, ·
(B.Maheshwar_C_program_Capstone.pdf	563 KB	May 14, 2025
(B.Maheshwar_C_program_Capstone(sample).pdf	563 KB	May 14, 2025
-\$NNY RDY PY.docx	162 B	May 29, 2025
2emu08064008.rar	3 MB	Feb 21, 2025
3CSA12_LabExperiments_2025_Template 2 Final.pptx	2007.0 KB	Mar 21, 2025
6x3 Tech Star summit 2025 Template 2 Final.pptx	1 MB	May 15, 2025
17647_chapter7-1.ppt	1007.5 KB	Feb 8, 2025
192411195_maheshwar.pdf	453.3 KB	May 15, 2025
192425259 G uday siva sai project 4.ppt	43.6 KB	Feb 20, 2025
192425259 G uday siva sai project 5.ppt	43.6 KB	Feb 20, 2025
192425259 G uday siva sai project 6.ppt	44.4 KB	Feb 20, 2025
192425259 G uday siva sai project 7.ppt	46.5 KB	Feb 20, 2025

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")

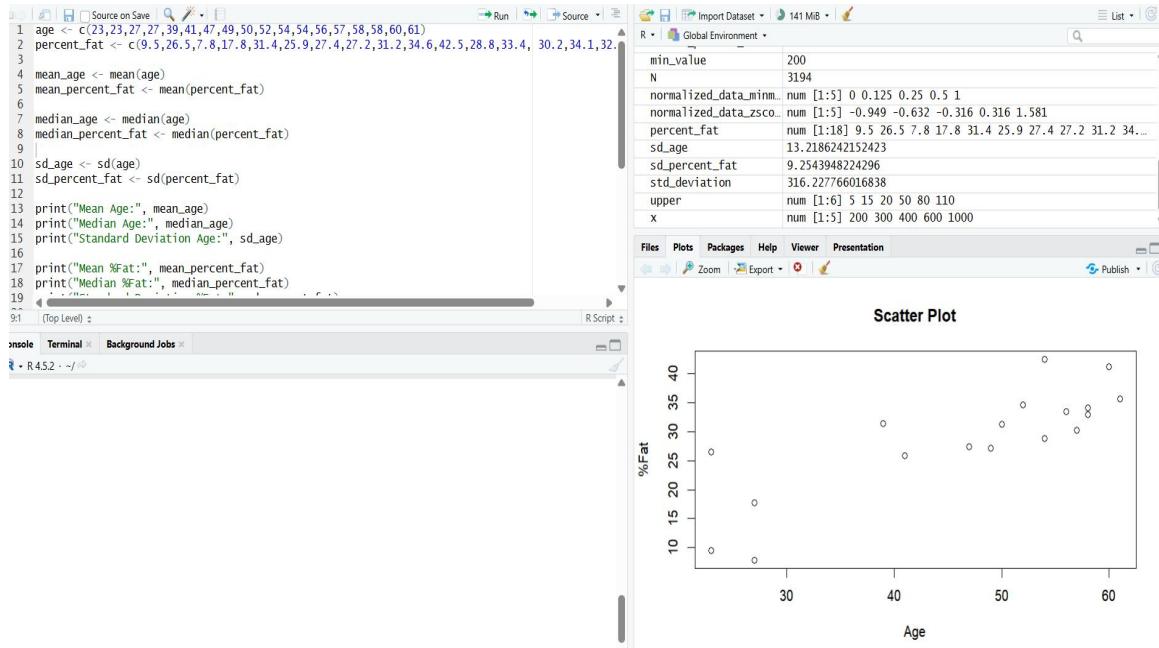
plot(age, percent_fat, main="Scatter Plot", xlab="Age", ylab="%Fat")

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:

The screenshot shows the RStudio interface with the following details:

- R Script:** The code provided in the question is pasted here.
- Console:** The output of the R code is displayed:

```
source("./dwdm_lab/exp6_..R")
[1] "Min-Max Normalized value: 35"
[2] "Z-Score Normalized value: 2.7047913446677"
[3] "Normalization by Decimal Scaling: 0.35"
```
- File Browser:** A sidebar showing the file structure and files in the current directory, including presentation files and documents.

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:

```
1 pencils <- c(9, 25, 23, 12, 11, 6, 7, 8, 9, 10)
2 mean_pencils <- mean(pencils)
3 median_pencils <- median(pencils)
4
5 get_mode <- function(x) {
6   ux <- unique(x)
7   ux[which.max(tabulate(match(x, ux)))]
8 }
9 mode_pencils <- get_mode(pencils)
10
11 print(paste("Mean:", mean_pencils))
12 print(paste("Median:", median_pencils))
13 print(paste("Mode:", mode_pencils))

[1] "Mean: 12"
[1] "Median: 9.5"
[1] "Mode: 9"
```

Program 8: Scatter Plot

Aim:

To draw scatter plot for given data.

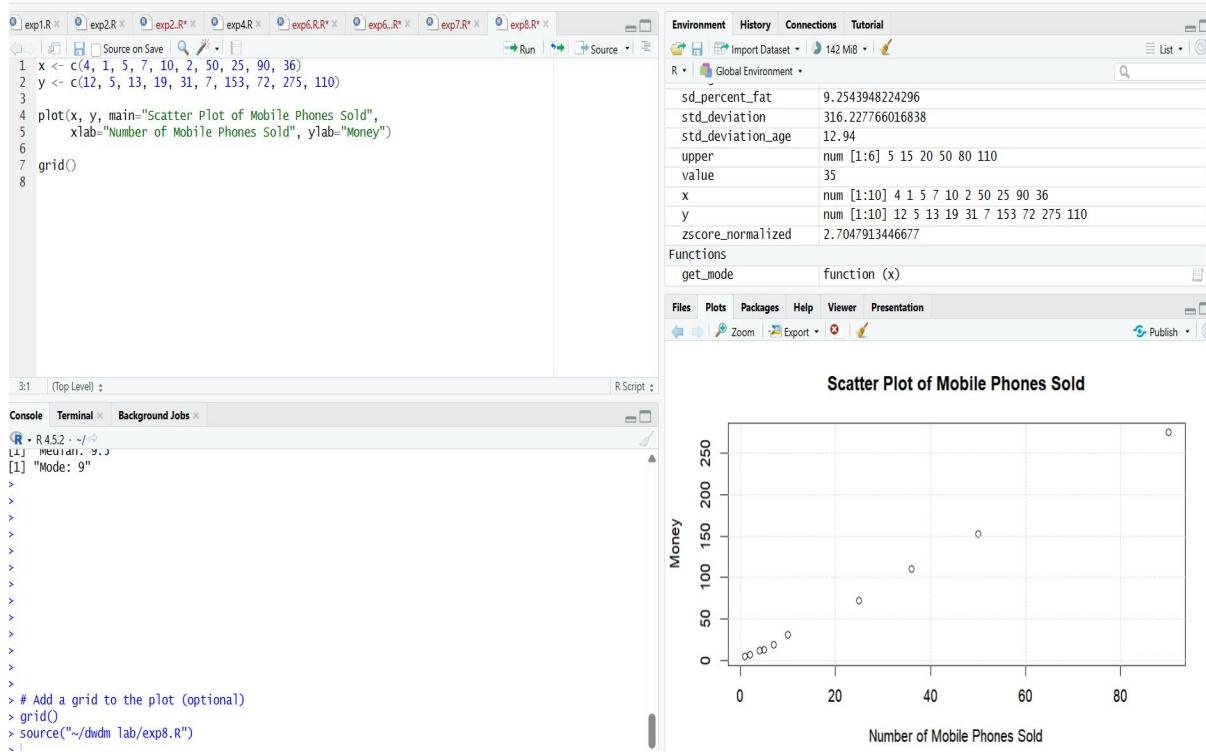
Algorithm:

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

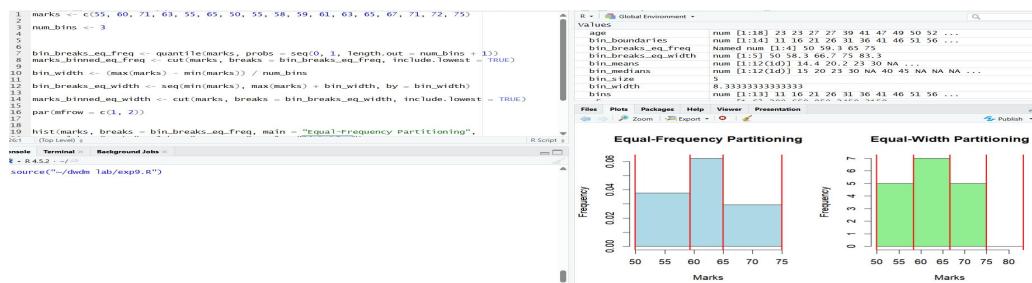
bin_breaks_eq_freq <- quantile(marks, probs = seq(0, 1, length.out = num_bins + 1))
marks_binned_eq_freq <- cut(marks, breaks = bin_breaks_eq_freq, include.lowest = TRUE)
bin_width <- (max(marks) - min(marks)) / num_bins
bin_breaks_eq_width <- seq(min(marks), max(marks) + bin_width, by = bin_width)
marks_binned_eq_width <- 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)

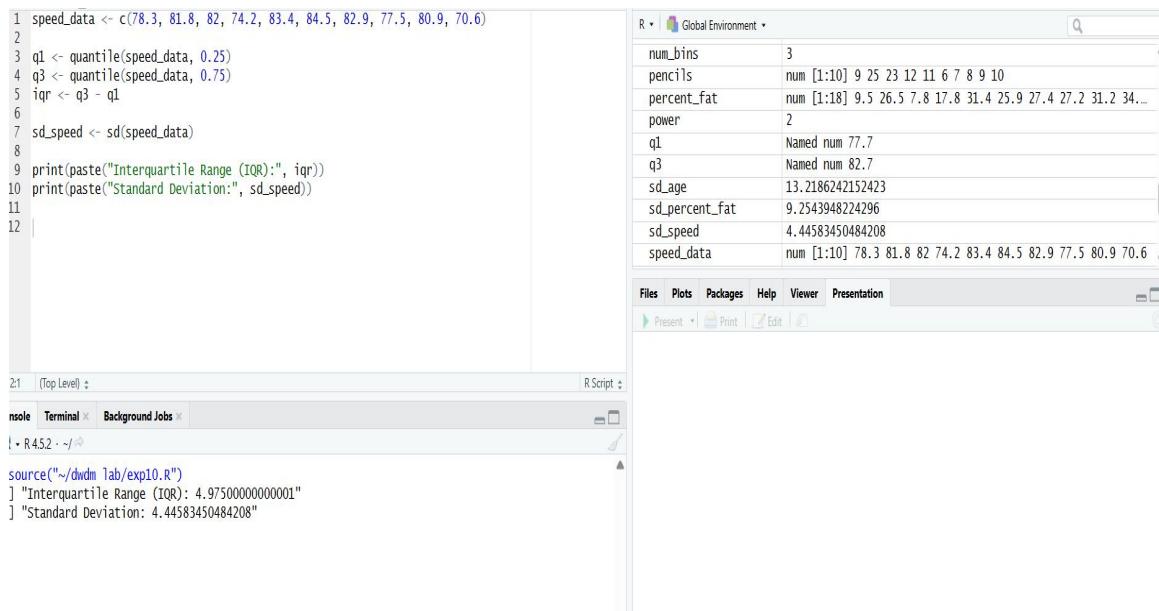
iqr <- q3 - q1

sd_speed <- sd(speed_data)

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

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

Outout:



The screenshot shows the RStudio interface with the code editor and global environment viewer.

Code Editor:

```
1 speed_data <- c(78.3, 81.8, 82, 74.2, 83.4, 84.5, 82.9, 77.5, 80.9, 70.6)
2
3 q1 <- quantile(speed_data, 0.25)
4 q3 <- quantile(speed_data, 0.75)
5 iqr <- q3 - q1
6
7 sd_speed <- sd(speed_data)
8
9 print(paste("Interquartile Range (IQR):", iqr))
10 print(paste("Standard Deviation:", sd_speed))
11
12 |
```

Global Environment:

Object	Type	Value
num_bins	integer	3
pencils	numeric	[1:10] 9 25 23 12 11 6 7 8 9 10
percent_fat	numeric	[1:18] 9.5 26.5 7.8 17.8 31.4 25.9 27.4 27.2 31.2 34.2 21.8 24.2 21.5 24.2 24.2 24.2 24.2 24.2
power	integer	2
q1	named numeric	77.7
q3	named numeric	82.7
sd_age	numeric	13.2186242152423
sd_percent_fat	numeric	9.2543948224296
sd_speed	numeric	4.44583450484208
speed_data	numeric	[1:10] 78.3 81.8 82 74.2 83.4 84.5 82.9 77.5 80.9 70.6

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, 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:

The screenshot shows an RStudio interface with the following details:

- Code Editor:** The left pane displays R code for calculating quartiles and standard deviation.
- Global Environment:** The right pane lists variables and their values.
- Console:** The bottom-left pane shows the output of the executed code.
- Toolbar:** The top bar includes tabs for Files, Plots, Packages, Help, Viewer, and Presentation.

```
1 age_data <- c(13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35)
2
3 total_data_points <- length(age_data)
4 q1_position <- (total_data_points + 1) / 4
5 q3_position <- 3 * q1_position
6
7 q1 <- age_data[ceiling(q1_position)]
8 q3 <- age_data[ceiling(q3_position)]
9
10 print(paste("Approximate First Quartile (Q1):", q1))
11 print(paste("Approximate Third Quartile (Q3):", q3))
12
```

	Global Environment
q1_position	7
q3	35
q3_position	21
sd_age	13.2186242152423
sd_percent_fat	9.2543948224296
sd_speed	4.44583450484208
speed_data	num [1:10] 78.3 81.8 82 74.2 83.4 ...
std_deviation	316.227766016838
std_deviation_age	12.94
total_data_points	27L

Files Plots Packages Help Viewer Presentation
Present Print Edit

5:32 (Top Level) ▾ R Script ▾

Console Terminal Background Jobs

R 4.5.2 - ~/

```
print(paste("Approximate Third Quartile (Q3):", q3))
[1] "Approximate Third Quartile (Q3): 82.675"
source("./dwdm_lab/exp_11.R")
[1] "Approximate First Quartile (Q1): 20"
```