

## DAY1: EXPERIMENTS

### So4

1.The intervals and corresponding frequencies are as follows. age frequency

1-5. 200

5-15 450

15-20 300

20-50 1500

50-80 700

80-110 44

Compute an approximate median value for the data

#### INPUT:

```
#age, frequency
```

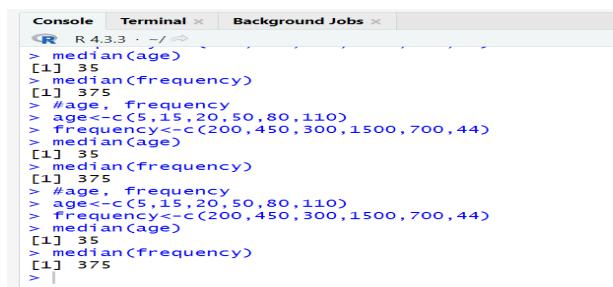
```
age<-c(5,15,20,50,80,110)
```

```
frequency<-c(200,450,300,1500,700,44)
```

```
median(age)
```

```
median(frequency)
```

#### OUTPUT:



```
Console Terminal × Background Jobs ×
R 4.3.3 · / ⌂
> median(age)
[1] 35
> median(frequency)
[1] 375
> #age, frequency
> age<-c(5,15,20,50,80,110)
> frequency<-c(200,450,300,1500,700,44)
> median(age)
[1] 35
> median(frequency)
[1] 375
> #age, frequency
> age<-c(5,15,20,50,80,110)
> frequency<-c(200,450,300,1500,700,44)
> median(age)
[1] 35
> median(frequency)
[1] 375
>
```

2.Suppose that the data for analysis includes the attribute age. The age values for the data tuples are (in increasing order) 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.

(a) What is the mean of the data? What is the median?

- (b) What is the mode of the data? Comment on the data's modality (i.e., bimodal, trimodal, etc.).
- (c) What is the midrange of the data?
- (d) Can you find (roughly) the first quartile (Q1) and the third quartile (Q3) of the data?

### Input:

```
#mean,median,mode,quatile

age<-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)

mean(age)

median(age)

mode_age<-names(table(age))[table(age)==max(table(age))]

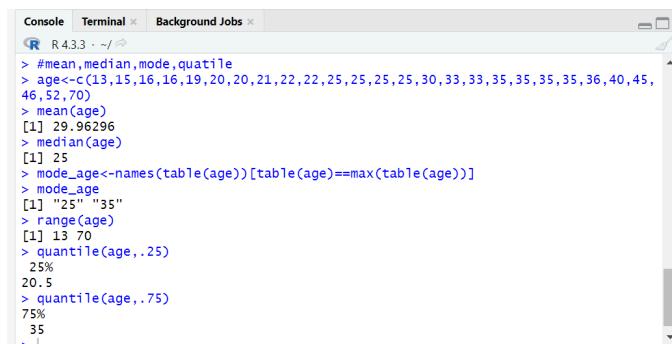
mode_age

range(age)

quantile(age,.25)

quantile(age,.75)
```

### OUTPUT:



A screenshot of an R console window titled "R 4.3.3". The console shows the following R code and its output:

```
Console Terminal Background Jobs
> #mean,median,mode,quatile
> age<-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)
> mean(age)
[1] 29.96296
> median(age)
[1] 25
> mode_age<-names(table(age))[table(age)==max(table(age))]
> mode_age
[1] "25" "35"
> range(age)
[1] 13 70
> quantile(age,.25)
25%
20.5
> quantile(age,.75)
75%
35
```

### 3.Data Preprocessing :Reduction and Transformation

Use the two methods below to normalize the following group of data: 200, 300, 400, 600, 1000  
 (a) min-max normalization by setting min = 0 and max = 1 (b) z-score normalization

### INPUT:

```
# Given data
```

```

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

# Min-Max normalization

min_max <- function(x) {

  (x - min(x)) / (max(x) - min(x))

}

min_max_normalized <- min_max(data)

print("Min-Max Normalized:")

print(min_max_normalized)

# Z-score normalization

z_score <- function(x) {

  (x - mean(x)) / sd(x)

}

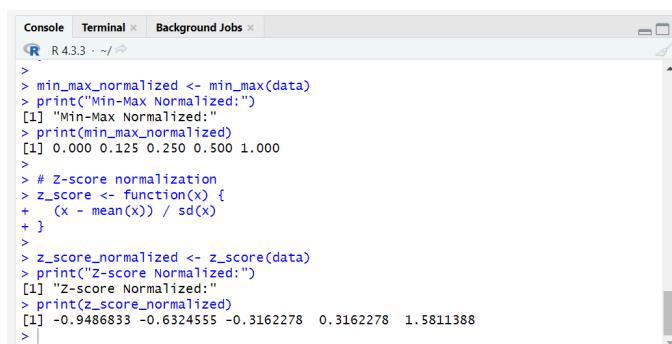
z_score_normalized <- z_score(data)

print("Z-score Normalized:")

print(z_score_normalized)

```

## OUTPUT:



The screenshot shows the RStudio interface with the 'Console' tab selected. The code has been run, and the output is displayed. The output shows the original data vector and the results of both min-max and z-score normalization.

```

Console Terminal Background Jobs
R 4.3.3 · ~/Documents

> data <- c(200, 300, 400, 600, 1000)
> min_max_normalized <- min_max(data)
> print("Min-Max Normalized:")
[1] "Min-Max Normalized:"
> print(min_max_normalized)
[1] 0.000 0.125 0.250 0.500 1.000
> 
> # Z-score normalization
> z_score <- function(x) {
+   (x - mean(x)) / sd(x)
+ }
> 
> z_score_normalized <- z_score(data)
> print("Z-score Normalized:")
[1] "Z-score Normalized:"
> print(z_score_normalized)
[1] -0.9486833 -0.6324555 -0.3162278  0.3162278  1.5811388
>

```

4..Data:11,13,13,15,15,16,19,20,20,20,21,21,22,23,24,30,40,45,45,45,71,  
 72,73,75 a) Smoothing by bin mean

b) Smoothing by bin median

c) Smoothing by bin boundaries

## Input:

```
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)
```

```
bins <- 5
```

```
bin_indices <- cut(data, bins)
```

```
mean_smooth <- tapply(data, bin_indices, mean)
```

```
print(mean_smooth)
```

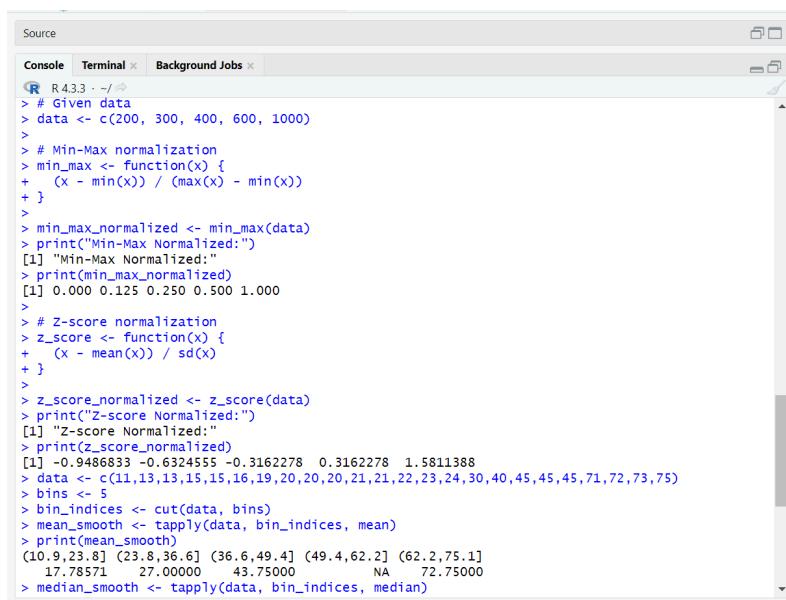
```
median_smooth <- tapply(data, bin_indices, median)
```

```
median_smooth
```

```
min_max_smooth <- tapply(data, bin_indices, function(x) c(min(x), max(x)))
```

```
print(min_max_smooth)
```

## OUTPUT:



The screenshot shows the RStudio interface with the 'Console' tab selected. The console window displays R code and its corresponding output. The code includes data generation, normalization functions (Min-Max and Z-score), binning, and smoothing operations. The output shows the generated data, the normalized data, and the resulting smoothed values.

```
R 4.3.3 - ~/ ◊
> # Given data
> data <- c(200, 300, 400, 600, 1000)
>
> # Min-Max normalization
> min_max <- function(x) {
+   (x - min(x)) / (max(x) - min(x))
+ }
>
> min_max_normalized <- min_max(data)
> print("Min-Max Normalized:")
[1] "Min-Max Normalized"
> print(min_max_normalized)
[1] 0.000 0.125 0.250 0.500 1.000
>
> # Z-score normalization
> z_score <- function(x) {
+   (x - mean(x)) / sd(x)
+ }
>
> z_score_normalized <- z_score(data)
> print("Z-score Normalized:")
[1] "Z-score Normalized"
> print(z_score_normalized)
[1] -0.9486833 -0.6324555 -0.3162278  0.3162278  1.5811388
> 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)
> bins <- 5
> bin_indices <- cut(data, bins)
> mean_smooth <- tapply(data, bin_indices, mean)
> print(mean_smooth)
[10.9,23.8] (23.8,36.6] (36.6,49.4] (49.4,62.2] (62.2,75.1]
 17.78571    27.00000   43.75000      NA    72.75000
> median_smooth <- tapply(data, bin_indices, median)
```

```

> median_smooth <- tapply(data, bin_indices, median)
> median_smooth
(10.9,23.8] (23.8,36.6] (36.6,49.4] (49.4,62.2] (62.2,75.1]
   19.5        27.0        45.0       NA        72.5
> min_max_smooth <- tapply(data, bin_indices, function(x) c(min(x), max(x)))
> print(min_max_smooth)
$ (10.9,23.8]
[1] 11 23

$ (23.8,36.6]
[1] 24 30

$ (36.6,49.4]
[1] 40 45

$ (49.4,62.2]
NULL

$ (62.2,75.1]
[1] 71 75

```

5. Suppose that a hospital tested the age and body fat data for 18 randomly selected adults with the following results:

<i>age</i>	23	23	27	27	39	41	47	49	50
%fat	9.5	26.5	7.8	17.8	31.4	25.9	27.4	27.2	31.2
<i>age</i>	52	54	54	56	57	58	58	60	61
%fat	34.6	42.5	28.8	33.4	30.2	34.1	32.9	41.2	35.7

- (a) Calculate the mean, median, and standard deviation of age and %fat.
- (b) Draw the boxplots for age and %fat.
- (c) Draw a scatter plot and a q-q plot based on these two variables.

### Input:

```

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

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)

median(age)

sd(age)

mean(fat)

median(fat)

sd(fat)

#boxplot

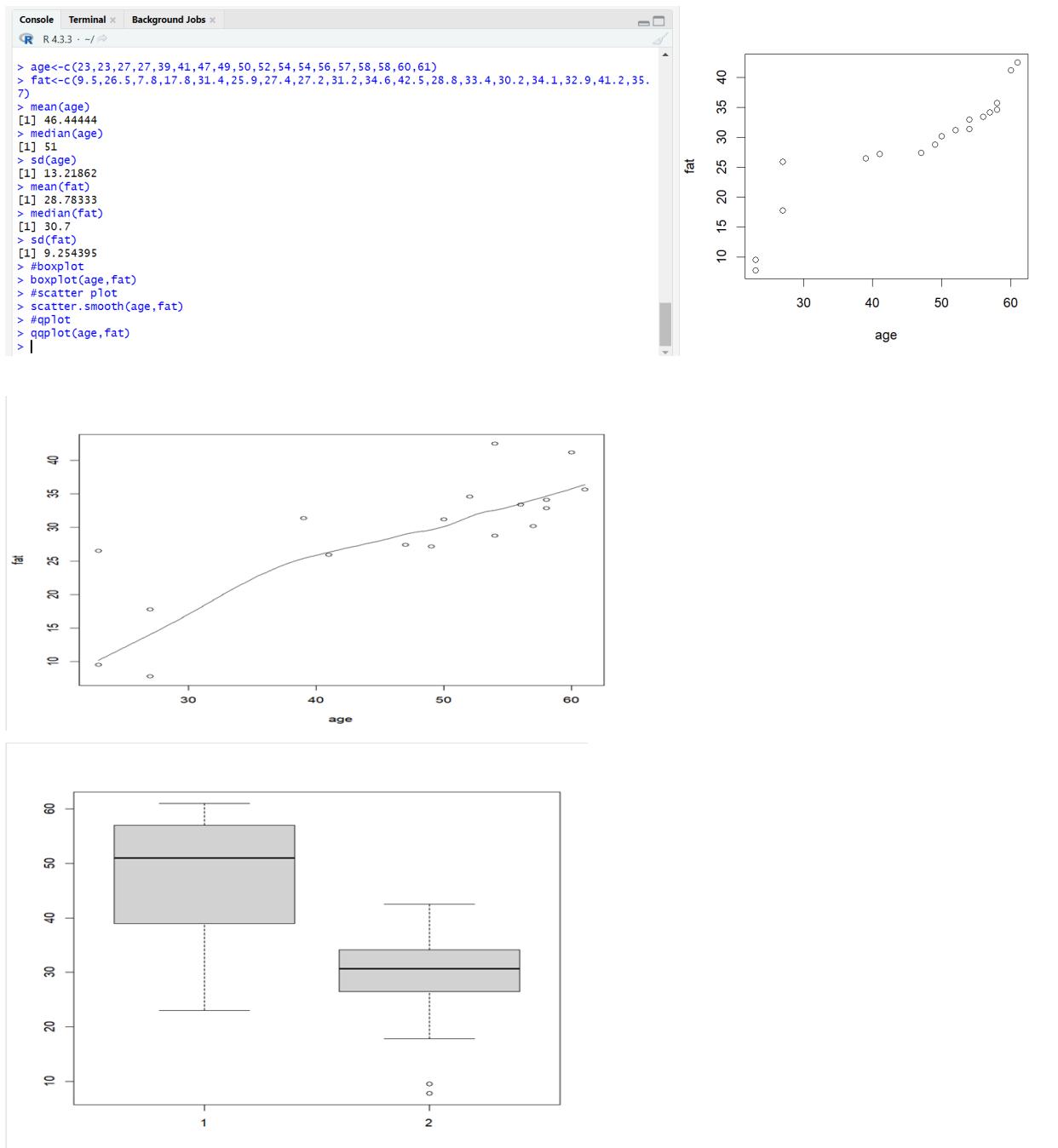
boxplot(age,fat)

```

```
#scatter plot
scatter.smooth(age,fat)

#qplot
qqplot(age,fat)
```

### OUTPUT:



6. Suppose that a hospital tested the age and body fat data for 18 randomly selected adults with the following results:

- (i) Use min-max normalization to transform the value 35 for age onto the range [0.0, 1.0].
- (ii) Use z-score normalization to transform the value 35 for age, where the standard deviation of age is 12.94 years.
- (iii) Use normalization by decimal scaling to transform the value 35 for age. Perform the above functions using R – tool

### **Input:**

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

```
min<-0
```

```
max<-1
```

```
#min_max
```

```
min_max=((35-min(v))/(max(v)-min(v)))
```

```
print(min_max)
```

```
#z-score
```

```
m=mean(v)
```

```
s<-12.94
```

```
z_score=(35-m)/s
```

```
print(z_score)
```

```
#decimal scaling
```

```
m<-35
```

```
j=max(m)<1
```

```
decimal_scaling=m/10^j
```

```
print(decimal_scaling)
```

### **OUTPUT:**

```

Console Terminal Background Jobs
R 4.3.3 · ~/Desktop

> age<-c(23,23,27,27,39,41,47,49,50,52,54,54,56,57,58,58,60,61)
> 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)
[1] 46.44444
> median(age)
[1] 51
> sd(age)
[1] 13.21862
> mean(fat)
[1] 28.78333
> median(fat)
[1] 30.7
> sd(fat)
[1] 9.254395
> #boxplot
> boxplot(age,fat)
> #scatter plot
> scatter.smooth(age,fat)
> #qplot
> qqplot(age,fat)
>

```

7. The following values are the number of pencils available in the different boxes. Create a vector and find out the mean, median and mode values of set of pencils in the given data.

Box1 Box2 Box3 Box4 Box5 Box6 Box7 Box8 Box9 Box 10

9        25        23        12        11        6        7        8        9        10

### Input:

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

mean(pencils)

median(pencils)

mode=names(table(pencils))[table(pencils)==max(table(pencils))]

### Mode

### OUTPUT:

```

Console Terminal Background Jobs
R 4.3.3 · ~/Desktop

> mean(pencils)
[1] 12
> median(pencils)
[1] 9.5
> mode=names(table(pencils))[table(pencils)==max(table(pencils))]
> mode
[1] "9"
>

```

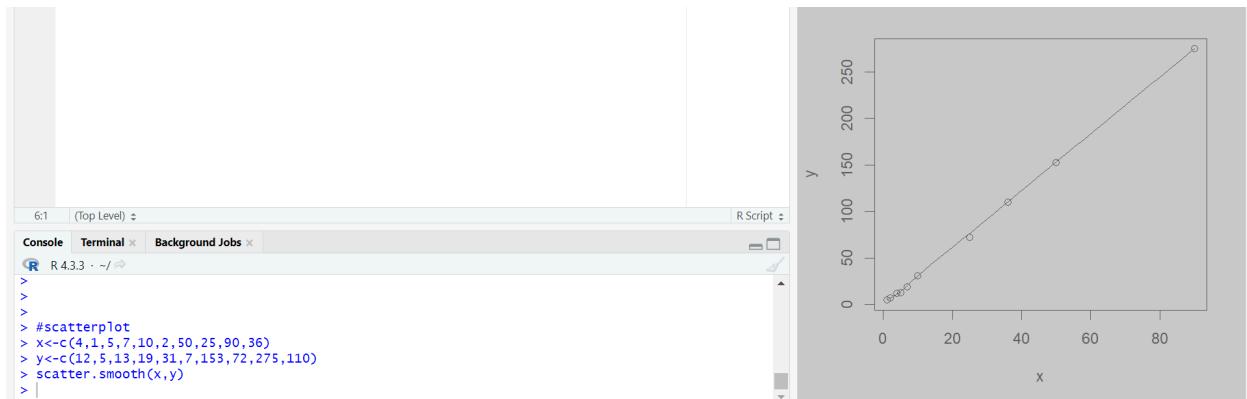
8. the following table would be plotted as (x,y) points, with the first column being the x values as number of mobile phones sold and the second column being the y values as money. To use the scatter plot for how many mobile phones sold.

```
x :4 1 5 7 10 2 50 25 90 36
```

```
y :12 5 13 19 31 7 153 72 275 110
```

**INPUT:**

```
#scatterplot  
x<-c(4,1,5,7,10,2,50,25,90,36)  
y<-c(12,5,13,19,31,7,153,72,275,110)  
scatter.smooth(x,y)
```

**OUTPUT:**

9. Implement of the R script using marks scored by a student in his model exam has been sorted as follows: 55, 60, 71, 63, 55, 65, 50, 55, 58, 59, 61, 63, 65, 67, 71, 72, 75. Partition them into three bins by each of the following methods. Plot the data points using histogram.

(a) equal-frequency (equi-depth) partitioning (b) equal-width partitioning

**Input:**

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

```
num_bins <- 3
```

```
bins_eq_frequency <- cut(marks, breaks = num_bins, labels = FALSE)
```

```
hist(marks, breaks = num_bins, col = "lightblue", xlab = "Marks", main =  
"Equal-Frequency (Equi-Depth) Partitioning")
```

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

```

bin_mean <- tapply(data, cut(data, num_bins), mean)

smoothed_data_by_mean <- unname(bin_mean[as.character(cut(data, num_bins))])

bin_median <- tapply(data, cut(data, num_bins), median)

smoothed_data_by_median <- unname(bin_median[as.character(cut(data, num_bins))])

bin_boundaries <- tapply(data, cut(data, num_bins), function(x) c(min(x), max(x)))

smoothed_data_by_boundaries <- unlist(bin_boundaries[as.character(cut(data, num_bins))])

print("Original data:")

print(data)

print("Smoothed data by bin mean:")

print(smoothed_data_by_mean)

print("Smoothed data by bin median:")

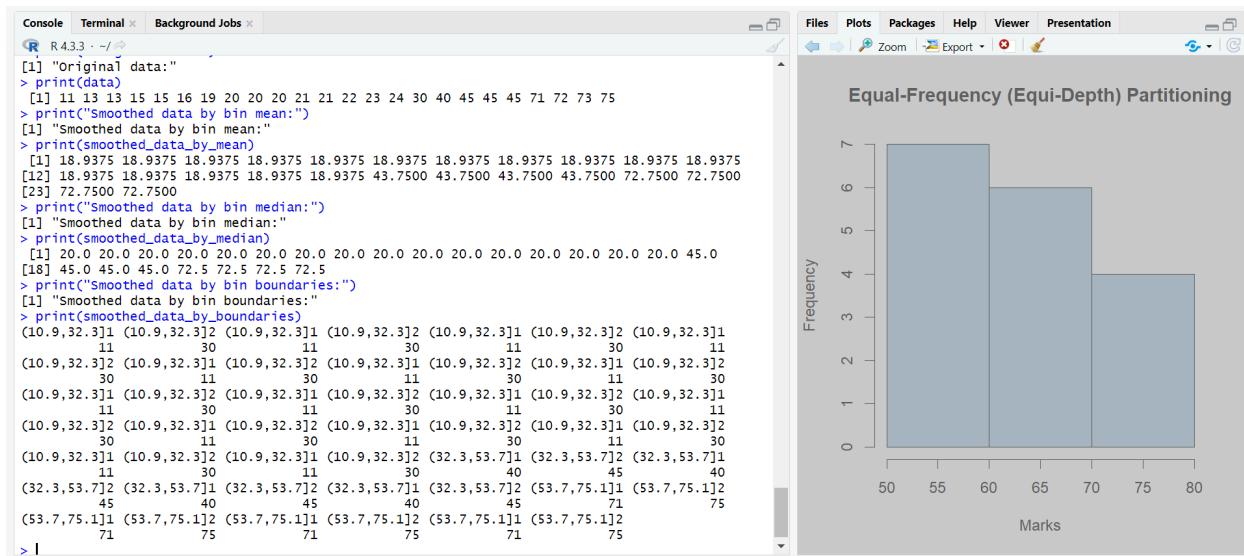
print(smoothed_data_by_median)

print("Smoothed data by bin boundaries:")

print(smoothed_data_by_boundaries)

```

## OUTPUT:



10. Suppose that the speed car is mentioned in different driving style.

Regular 78.3 81.8 82 74.2 83.4 84.5 82.9 77.5 80.9 70.6 Speed

Calculate the Inter quantile and standard deviation of the given data.

### Input:

#IQR, SD

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

IQR(v)

sd(v)

### OUTPUT:



The screenshot shows the RStudio interface with the 'Console' tab selected. The R version is R 4.3.3. The session history shows the following commands and their results:

```
R 4.3.3 --> v<-c(78.3,81.8,82,74.2,83.4,84.5,82.9,77.5,80.9,70.6)
R 4.3.3 --> IQR(v)
[1] 4.975
R 4.3.3 --> sd(v)
[1] 4.445835
```

11. Suppose that the data for analysis includes the attribute age. The age values for the data tuples are (in increasing order) 13, 15, 16, 16, 16, 19, 20, 20, 20, 21, 22, 22, 22, 25, 25, 25, 25, 30, 33, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70.

Can you find (roughly) the first quartile (Q1) and the third quartile (Q3) of the data?

### Input:

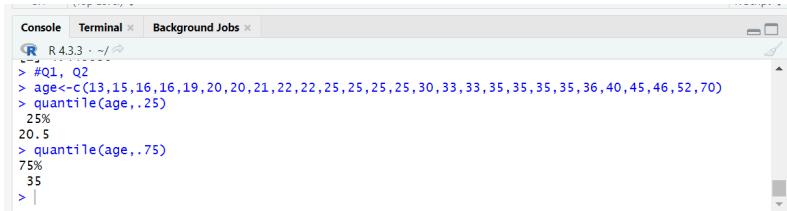
#Q1, Q2

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

quantile(age,.25)

quantile(age,.75)

### OUTPUT:



```
R 4.3.3 : ~/ ◊
> #Q1, Q2
> age<-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)
> quantile(age,.25)
25%
20.5
> quantile(age,.75)
75%
35
> |
```

## DAY 2: EXPERIMENTS

### 1. Covariance and correlation

Children of three ages are asked to indicate their preference for three photographs of adults. Do the data suggest that there is a significant relationship between age and photograph preference? What is wrong with this study?

#### Photograph:

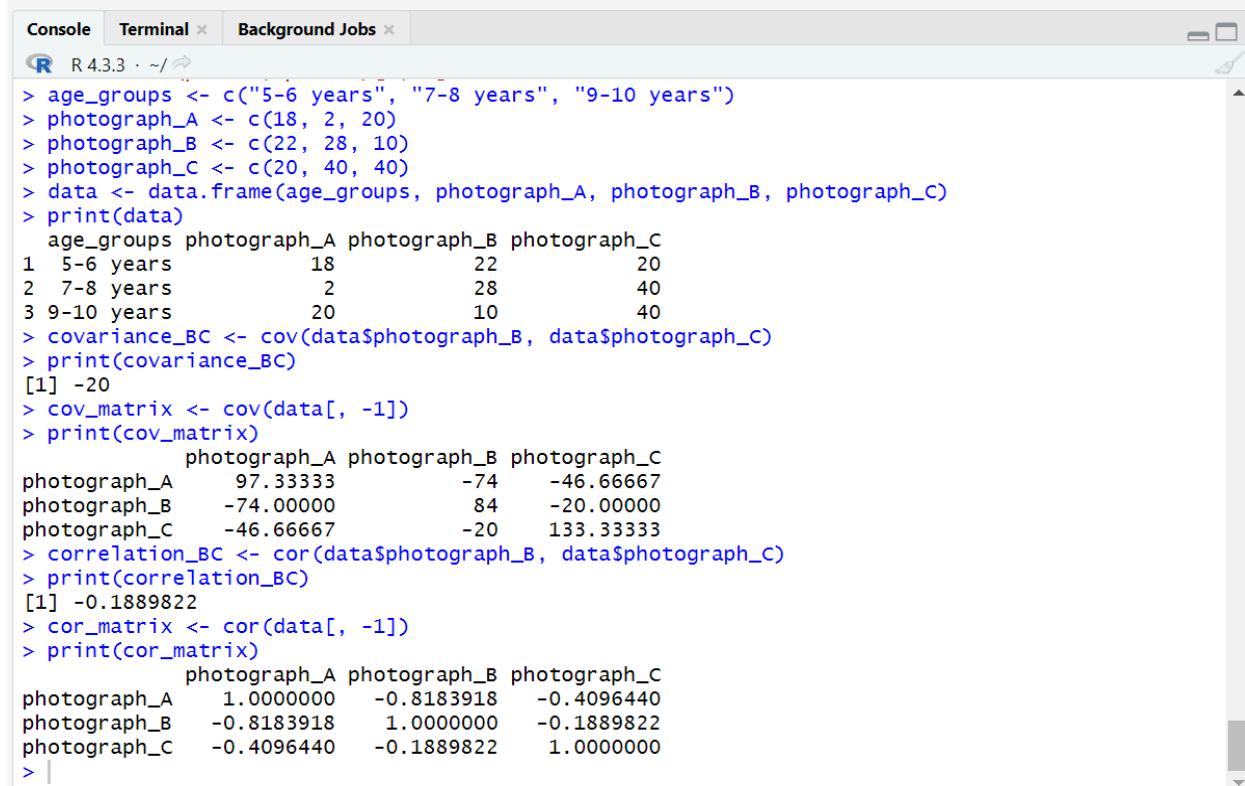
Age of child	A	B	C
5-6 years:	18	22	20
7-8 years:	2	28	40
9-10 years:	20	10	40

1. Use `cov()` to calculate the sample covariance between B and C.
2. Use another call to `cov()` to calculate the sample covariance matrix for the preferences.
3. Use `cor()` to calculate the sample correlation between B and C.
4. Use another call to `cor()` to calculate the sample correlation matrix for the preferences.

INPUT:

```
age_groups <- c("5-6 years", "7-8 years", "9-10 years")
photograph_A <- c(18, 2, 20)
photograph_B <- c(22, 28, 10)
photograph_C <- c(20, 40, 40)
data <- data.frame(age_groups, photograph_A, photograph_B, photograph_C)
print(data)
covariance_BC <- cov(data$photograph_B, data$photograph_C)
print(covariance_BC)
cov_matrix <- cov(data[, -1])
print(cov_matrix)
correlation_BC <- cor(data$photograph_B, data$photograph_C)
print(correlation_BC)
cor_matrix <- cor(data[, -1])
print(cor_matrix)
```

OUTPUT:



The screenshot shows the RStudio interface with the 'Console' tab selected. The console window displays the R code and its corresponding output. The code defines variables for age groups and photograph counts, creates a data frame, and performs covariance and correlation calculations between photograph\_B and photograph\_C. The output shows the data frame, covariance matrix, and correlation matrix.

```
R 4.3.3 · ~/Documents/RStudioProjects/Untitled Project
Console Terminal × Background Jobs ×
> age_groups <- c("5-6 years", "7-8 years", "9-10 years")
> photograph_A <- c(18, 2, 20)
> photograph_B <- c(22, 28, 10)
> photograph_C <- c(20, 40, 40)
> data <- data.frame(age_groups, photograph_A, photograph_B, photograph_C)
> print(data)
  age_groups photograph_A photograph_B photograph_C
1 5-6 years           18          22         20
2 7-8 years            2          28         40
3 9-10 years           20          10         40
> covariance_BC <- cov(data$photograph_B, data$photograph_C)
> print(covariance_BC)
[1] -20
> cov_matrix <- cov(data[, -1])
> print(cov_matrix)
            photograph_A photograph_B photograph_C
photograph_A    97.33333     -74      -46.66667
photograph_B   -74.00000      84      -20.00000
photograph_C   -46.66667     -20     133.33333
> correlation_BC <- cor(data$photograph_B, data$photograph_C)
> print(correlation_BC)
[1] -0.1889822
> cor_matrix <- cor(data[, -1])
> print(cor_matrix)
            photograph_A photograph_B photograph_C
photograph_A    1.0000000   -0.8183918   -0.4096440
photograph_B   -0.8183918    1.0000000   -0.1889822
photograph_C   -0.4096440   -0.1889822    1.0000000
> |
```

2. Imagine that you have selected data from the All Electronics data warehouse for analysis. The data set will be huge! The following data are a list of All Electronics prices for commonly sold items (rounded to the nearest dollar). The numbers have been

sorted: 1, 1, 5, 5, 5, 5, 5, 8, 8, 10, 10, 10, 10, 12, 14, 14, 14, 15, 15, 15, 15, 15, 15, 15, 15, 18, 18, 18, 18, 18, 8, 8, 10, 10, 10, 12, 14, 14, 14, 15, 15, 15, 15, 15, 15, 18, 18, 18, 18,

18, 18, 18, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 25, 25, 25, 25, 25, 25, 28, 28, 30,

30, 30.

- (i) Partition the dataset using an equal-frequency partitioning method with bin equal to 3
- (ii) apply data smoothing using bin means and bin boundary.
- (iii) Plot Histogram for the above frequency division

INPUT:

# Define the dataset

```
prices <- c(1, 1, 5, 5, 5, 5, 5, 8, 8, 10, 10, 10, 10, 12, 14, 14, 14, 15, 15, 15, 15, 15, 15, 15, 18, 18, 18, 18, 18, 8, 8, 10, 10, 10, 10, 12, 14, 14, 14, 15, 15, 15, 15, 15, 15, 18, 18, 18, 18, 18, 18, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 25, 25, 25, 25, 25, 25, 28, 28, 30, 30, 30)
```

# (i) Partition the dataset using equal-frequency partitioning with bin equal to 3  
partitioned <- cut(prices, breaks = 3, labels = FALSE)

# (ii) Apply data smoothing using bin means  
bin\_means <- tapply(prices, partitioned, mean)  
smoothed\_prices <- bin\_means[partitioned]

# (iii) Plot Histogram  
hist(prices, breaks = 3, main = "Histogram of Prices", xlab = "Price", col = "skyblue", border = "black")

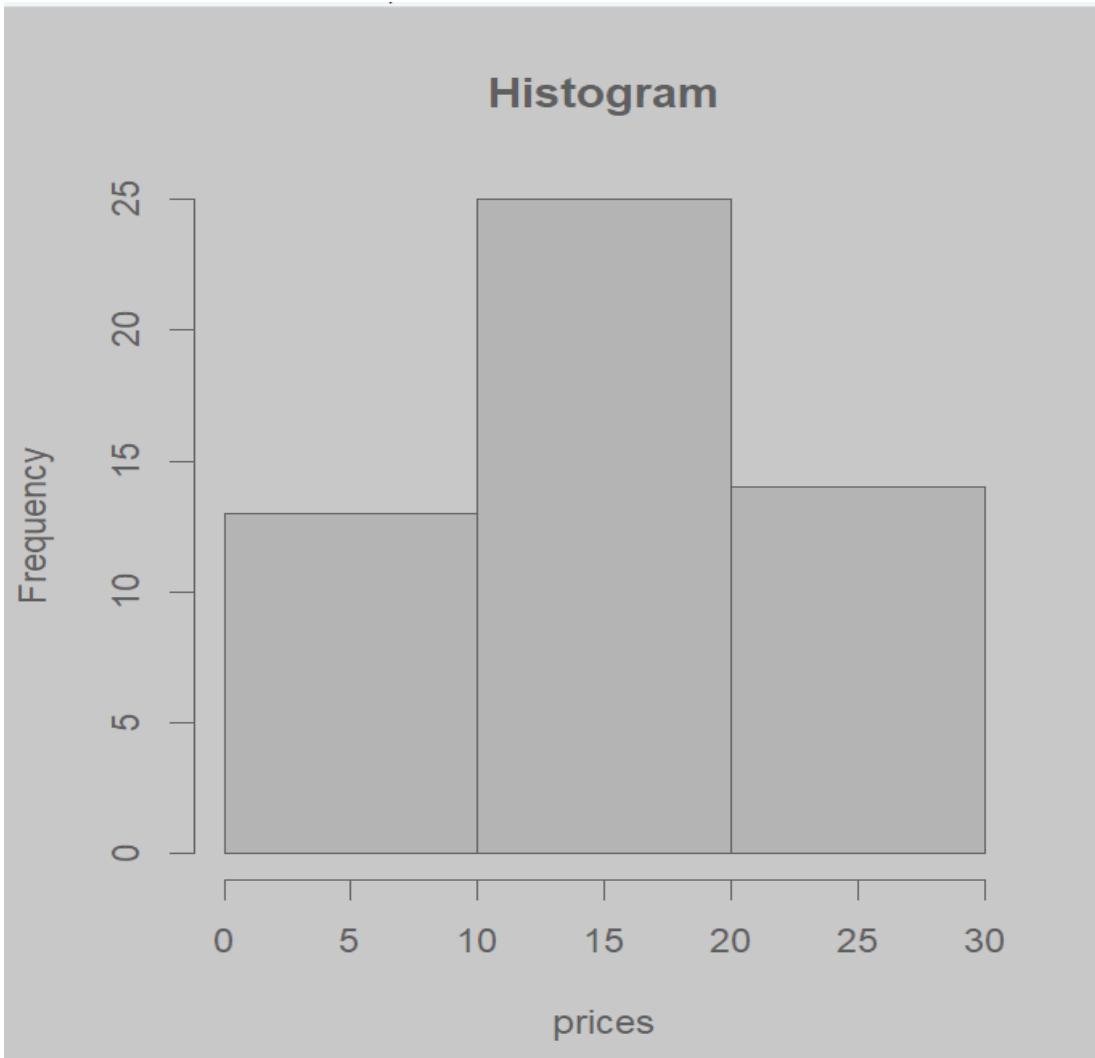
```
Console Terminal x Background Jobs x
R 4.3.3 · ~/ 
> bin<-length(data)/3
> bins<-cut(data, breaks = c(-Inf, quantile(data, probs = seq(0,1, 1/3)),Inf), include.lowest = TRUE)
> tapply(data,bins,mean)
[1,-Inf,1]      (1,15]     (15,20]    (20,30]   (30, Inf]
1.000000 10.71429 18.93333 25.35714 NA
> tapply(data, bins,function(x) c(min(x),max(x)))
$`[-Inf,1]` 
[1] 1 1

$`(1,15]` 
[1] 5 15

$`(15,20]` 
[1] 18 20

$`(20,30]` 
[1] 21 30

$`(30, Inf]` 
NULL
```



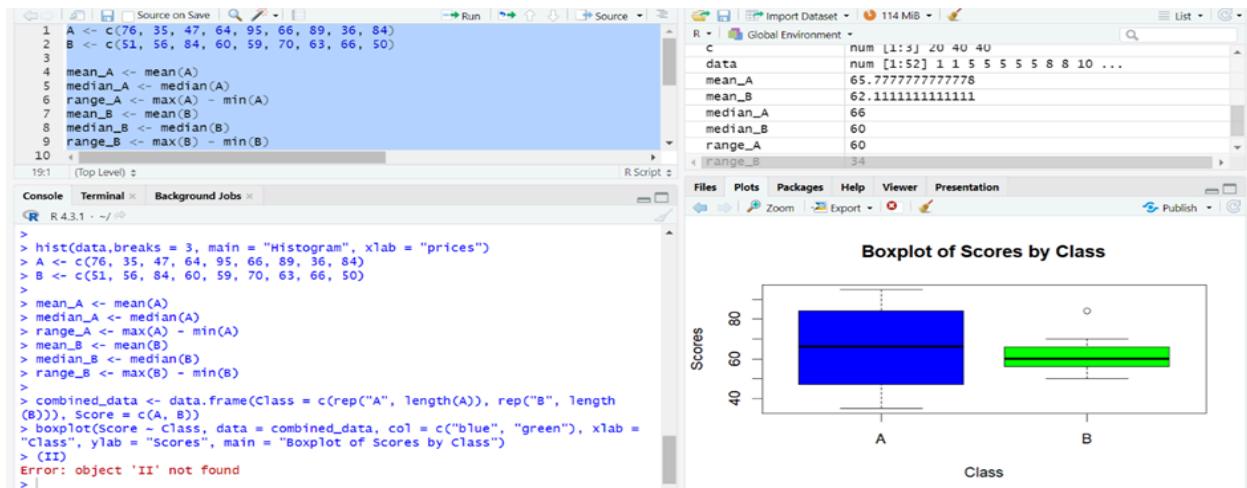
3.Two Maths teachers are comparing how their Year 9 classes performed in the end of year exams. Their results are as follows:

Class A: 76, 35, 47, 64, 95, 66, 89, 36, 84, 76, 35, 47, 64, 95, 66, 89, 36, 84

Class B: 51, 56, 84, 60, 59, 70, 63, 66, 50, 51, 56, 84, 60, 59, 70, 63, 66, 50

(i) Find which class had scored higher mean, median and range.

(ii) Plot above in boxplot and give the inferences



4.Let us consider one example to make the calculation method clear. Assume that the minimum and maximum values for the feature F are \$50,000 and \$100,000 correspondingly. It needs to range F from 0 to 1. In accordance with min-max normalization,  $v = \$80$ ,

b) Use the two methods below to normalize the following group of data: 200, 300, 400, 600, 1000

(a) min-max normalization by setting min = 0 and max = 1

(b) z-score normalization

INPUT:

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

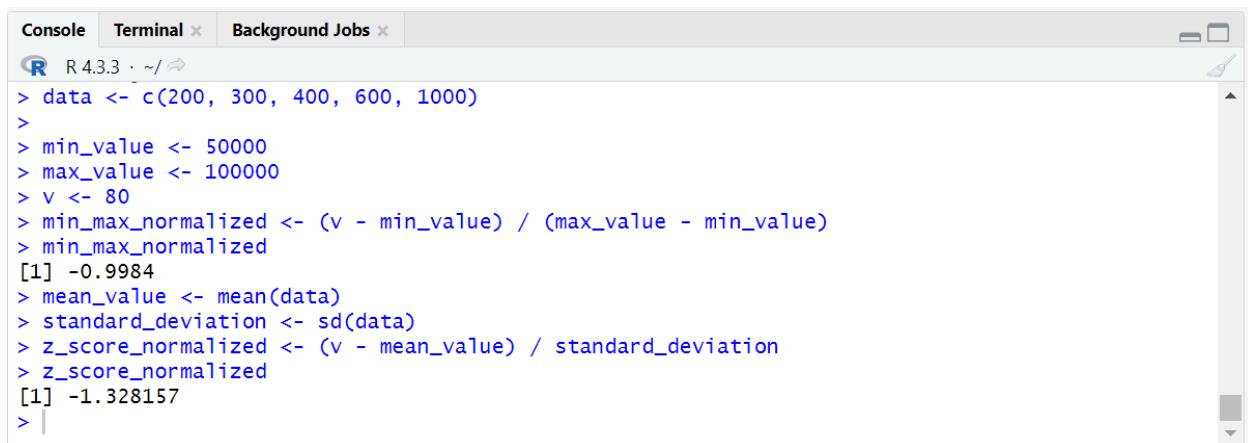
min\_value <- 50000

```

max_value <- 100000
v <- 80
min_max_normalized <- (v - min_value) / (max_value - min_value)
min_max_normalized
mean_value <- mean(data)
standard_deviation <- sd(data)
z_score_normalized <- (v - mean_value) / standard_deviation
z_score_normalized

```

OUTPUT:



The screenshot shows the RStudio interface with the 'Console' tab selected. The console window displays the following R code and its output:

```

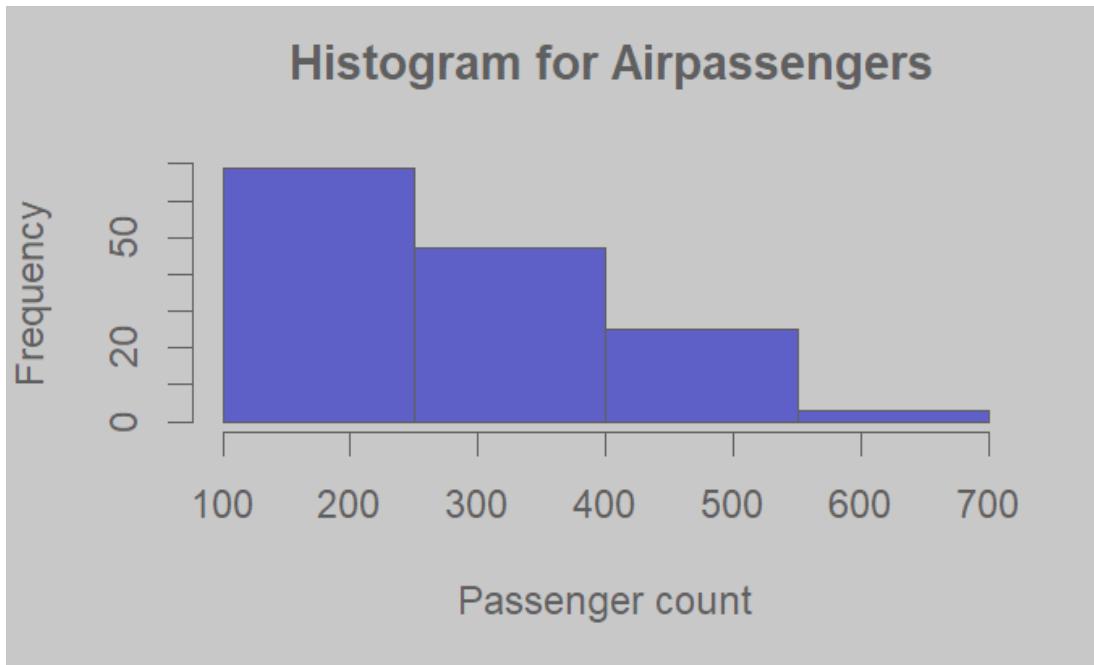
Console Terminal × Background Jobs ×
R 4.3.3 · ~/r
> data <- c(200, 300, 400, 600, 1000)
>
> min_value <- 50000
> max_value <- 100000
> v <- 80
> min_max_normalized <- (v - min_value) / (max_value - min_value)
> min_max_normalized
[1] -0.9984
> mean_value <- mean(data)
> standard_deviation <- sd(data)
> z_score_normalized <- (v - mean_value) / standard_deviation
> z_score_normalized
[1] -1.328157
> |

```

5. Make a histogram for the “AirPassengers” dataset, start at 100 on the x-axis, and from values 200 to 700, make the bins 150 wide

INPUT:

```
Console Terminal x Background Jobs x
R 4.3.3 · ~/🔗
> data <- c(200, 300, 400, 600, 1000)
>
> min_value <- 50000
> max_value <- 100000
> v <- 80
> min_max_normalized <- (v - min_value) / (max_value - min_value)
> min_max_normalized
[1] -0.9984
> mean_value <- mean(data)
> standard_deviation <- sd(data)
> z_score_normalized <- (v - mean_value) / standard_deviation
> z_score_normalized
[1] -1.328157
> data("AirPassengers")
> hist(AirPassengers, breaks = seq(100, 700, by = 150), col = "blue", main=" Histogram for Airpassengers", xlab = "Passenger count", ylab = "Frequency")
> |
```



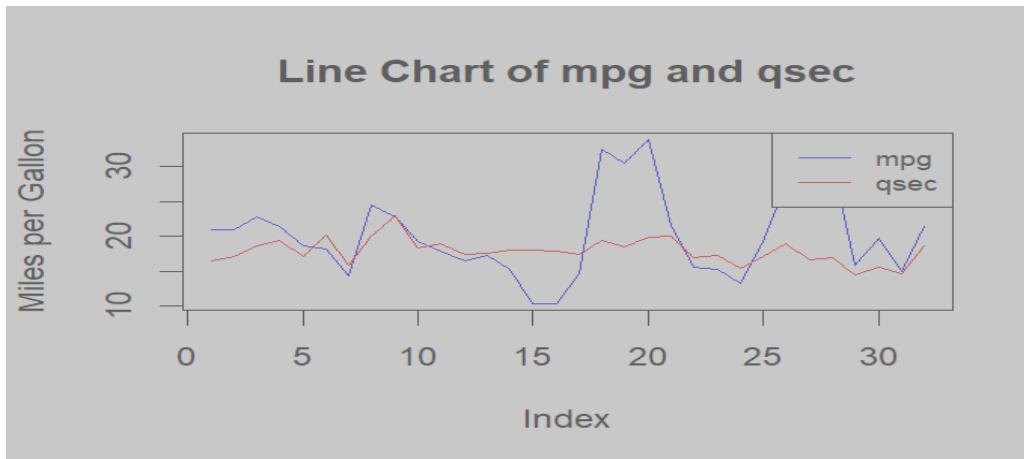
6. Obtain Multiple Lines in Line Chart using a single Plot Function in R. Use attributes "mpg" and "qsec" of the dataset "mtcars"

INPUT:

```
data("mtcars")
plot(mtcars$mpg, type = "l", col = "blue", xlab = "Index", ylab = "Miles per Gallon", main = "Line Chart of mpg and qsec")
lines(mtcars$qsec, col = "red")
```

```
legend("topright", legend = c("mpg", "qsec"), col = c("blue", "red"), lty = 1, cex = 0.8)
```

OUTPUT:



7.Download the Dataset "water" From R dataset Link.Find out whether there is a linear relation between attributes"mortality" and"hardness" by plot function.Fit the Data into the Linear Regression model.Predict the mortality for the hardness=88.

INPUT:

```
data("iris")
str(iris)

plot(iris$Sepal.Length, iris$Petal.Length, main = "Scatter plot of Sepal.Length vs.
Petal.Length", xlab = "Sepal.Length", ylab = "Petal.Length", col = "blue", pch = 16)

model <- lm(Petal.Length ~ Sepal.Length, data = iris)

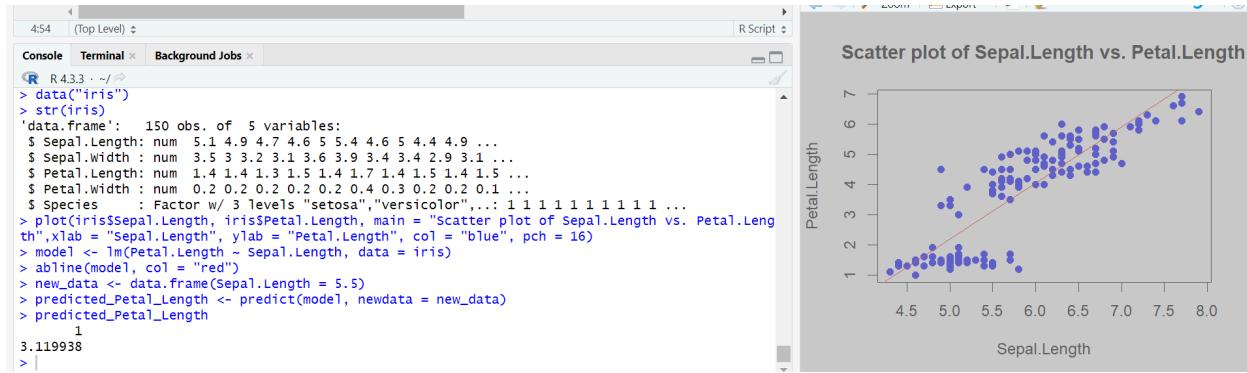
abline(model, col = "red")

new_data <- data.frame(Sepal.Length = 5.5)

predicted_Petal_Length <- predict(model, newdata = new_data)

predicted_Petal_Length
```

OUTPUT:



8.Create a Boxplot graph for the relation between "mpg"(miles per gallon) and "cyl"(number of Cylinders) for the dataset "mtcars" available in R Environment.

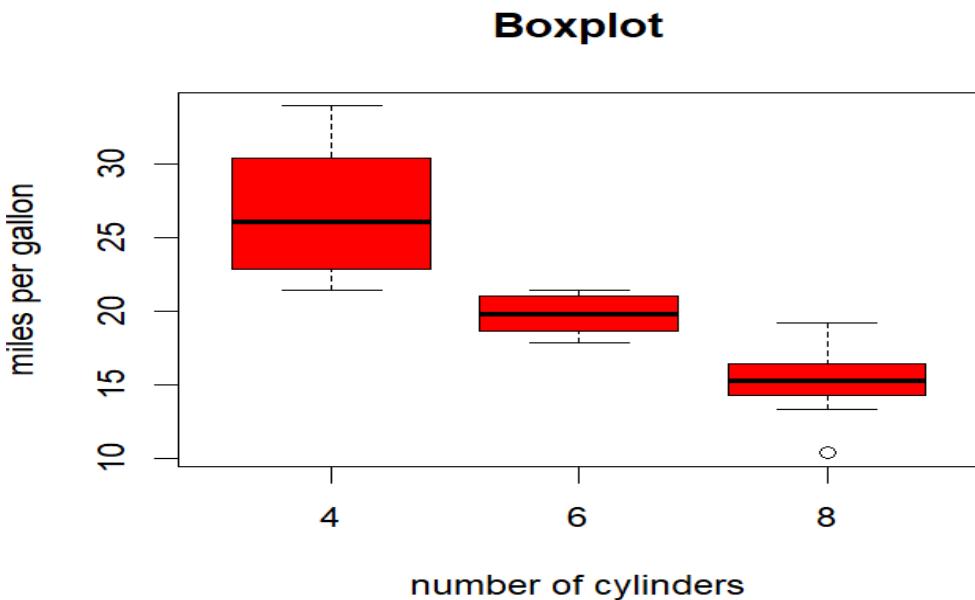
**INPUT:**

```

data("mtcars") 
boxplot(mpg ~ cyl, data = mtcars, main = "Boxplot", xlab = "number of cylinders", ylab = "miles per gallon", col= "red")

```

**OUTPUT:**



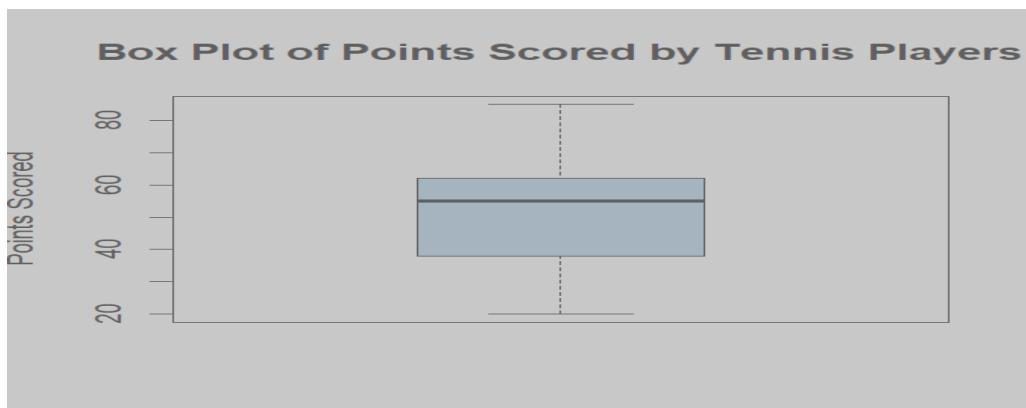
9. Assume the Tennis coach wants to determine if any of his team players are scoring outliers. To visualize the distribution of points scored by his players, then how can he decide to develop the box plot? Give suitable example using Boxplot visualization technique.

INPUT:

```
score <- c(20, 25, 30, 32, 35, 38, 40, 45, 50, 52, 55, 56, 58, 59, 60, 62, 65, 70, 75, 80, 85)
```

```
boxplot(score, col = "lightblue", main = "Box Plot of Points Scored by Tennis Players", ylab = "Points Scored")
```

OUTPUT:



10. Implement using R language in which age group of people are affected by blood pressure based on the diabetes dataset show it using scatter plot and bar chart (that is BloodPressure vs Age using dataset “diabetes.csv”)

INPUT:

```
# Load the dataset  
  
diabetes_data <- read.csv("C:/Users/V.POOGITHA/Downloads/diabetes.csv")  
  
# Create a scatterplot  
  
plot(diabetes_data$Age, diabetes_data$BloodPressure,  
      main = "Blood Pressure vs. Age in Sample Diabetes Data",  
      xlab = "Age", ylab = "Blood Pressure")  
  
# Create age groups (adjust as needed)  
  
age_groups <- cut(diabetes_data$Age, breaks = c(0, 30, 40, 50, 60, Inf))
```

```

# Calculate mean blood pressure for each age group

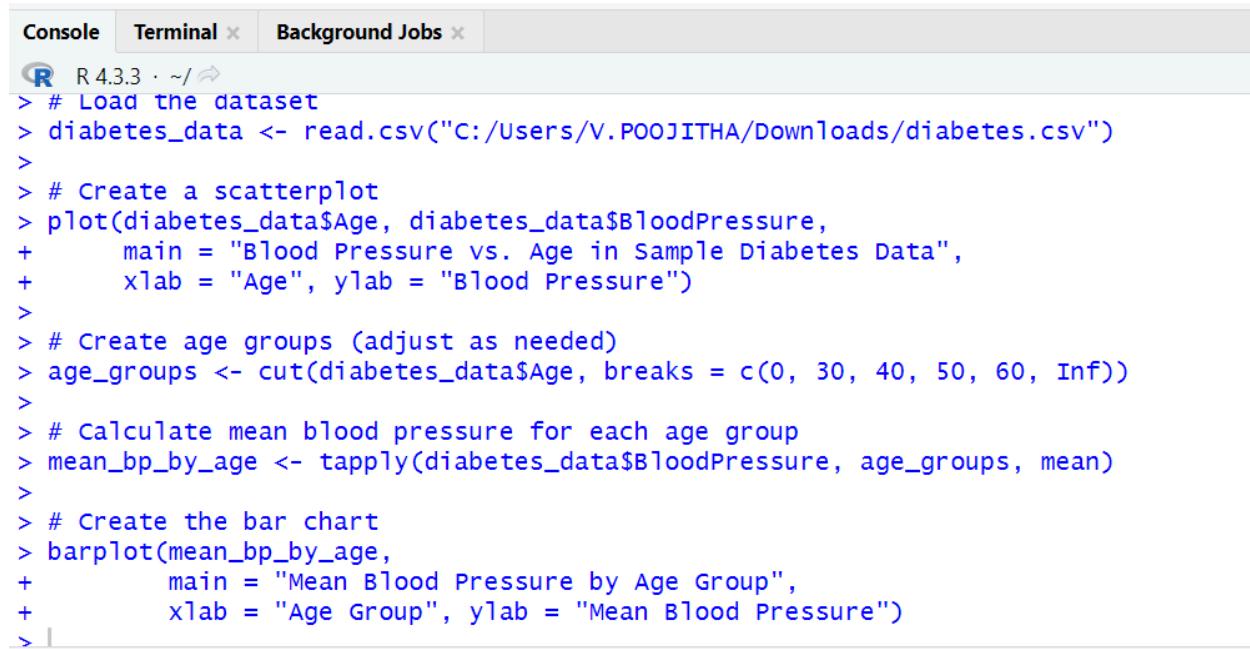
mean_bp_by_age <- tapply(diabetes_data$BloodPressure, age_groups, mean)

# Create the bar chart

barplot(mean_bp_by_age,
        main = "Mean Blood Pressure by Age Group",
        xlab = "Age Group", ylab = "Mean Blood Pressure")

```

OUTPUT:

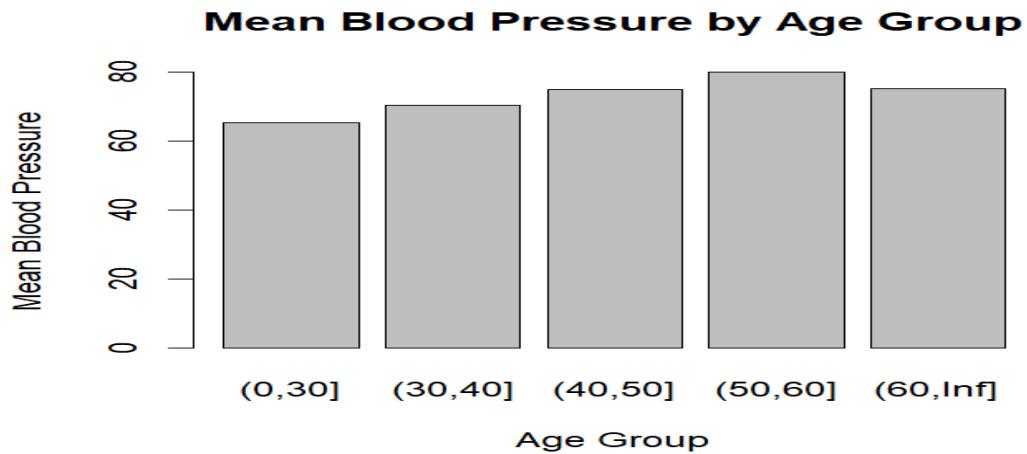


The screenshot shows the RStudio interface with the 'Console' tab selected. The R session window displays the following code:

```

Console Terminal x Background Jobs x
R 4.3.3 · ~/ ↗
> # Load the dataset
> diabetes_data <- read.csv("C:/Users/V.POJJITHA/Downloads/diabetes.csv")
>
> # Create a scatterplot
> plot(diabetes_data$Age, diabetes_data$BloodPressure,
+       main = "Blood Pressure vs. Age in Sample Diabetes Data",
+       xlab = "Age", ylab = "Blood Pressure")
>
> # Create age groups (adjust as needed)
> age_groups <- cut(diabetes_data$Age, breaks = c(0, 30, 40, 50, 60, Inf))
>
> # Calculate mean blood pressure for each age group
> mean_bp_by_age <- tapply(diabetes_data$BloodPressure, age_groups, mean)
>
> # Create the bar chart
> barplot(mean_bp_by_age,
+           main = "Mean Blood Pressure by Age Group",
+           xlab = "Age Group", ylab = "Mean Blood Pressure")
>

```



### DAY 3: EXPERIMENTS

1. Consider the data set and perform the Apriori Algorithm and FP algorithm support:3 and confidence=50%

Customer ID	Transaction ID	Items Bought
1	0001	{a, d, e}
1	0024	{a, b, c, e}
2	0012	{a, b, d, e}
2	0031	{a, c, d, e}
3	0015	{b, c, e}
3	0022	{b, d, e}
4	0029	{c, d}
4	0040	{a, b, c}
5	0033	{a, d, e}
5	0038	{a, b, e}

Input:

@relation dataset

@attribute a{true,false}

@attribute b{true,false}

```
@attribute c{true,false}
```

```
@attribute d{true,false}
```

```
@attribute e{true,false}
```

```
@data
```

```
true false false true true
```

```
true true true false true
```

```
true true false true true
```

```
true false true true true
```

```
false true true false true
```

```
false true false true true
```

```
false false true true false
```

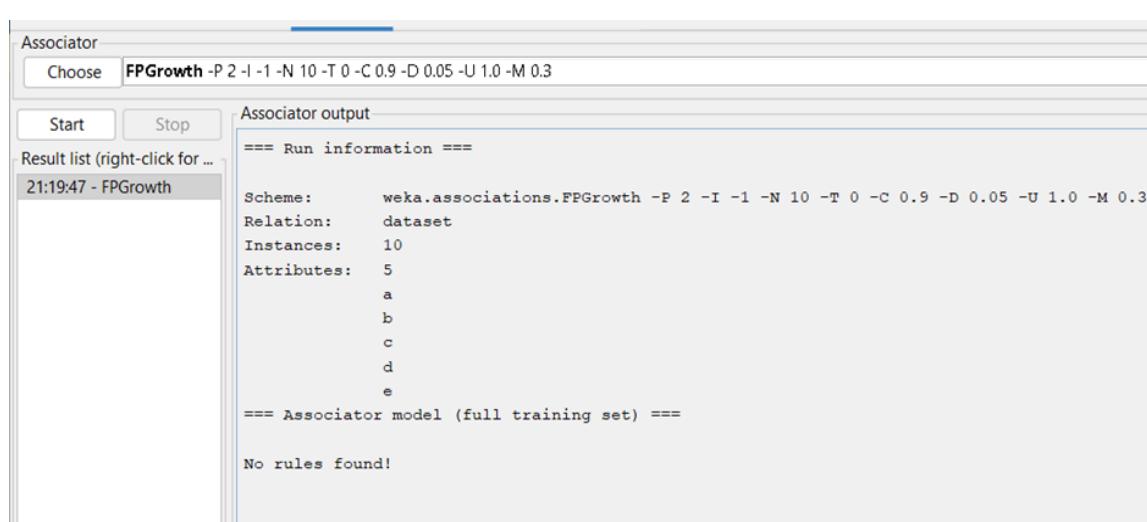
```
true true true false false
```

```
true false false true true
```

```
true true false false true
```

### Output:

### FP GROWTH:



## APRIORI ALGORITHM:

The screenshot shows the 'Associator' software interface. In the top bar, the command 'Apriori -N 10 -T 0 -C 0.5 -D 0.05 -U 1.0 -M 0.3 -S -1.0 -c -1' is selected. Below the bar are 'Start' and 'Stop' buttons. A 'Result list' window on the left shows '21:19:47 - FP-Growth' and '21:38:17 - Apriori' (which is currently selected). The main window displays the 'Associator output' for the Apriori model. It includes parameters: Minimum support: 0.45 (4 instances), Minimum metric <confidence>: 0.5, and Number of cycles performed: 11. It lists generated itemsets L(1) through L(3) and their sizes: 8, 10, and 3 respectively. The best rules found are:

1. c=false 5 ==> e=true 5      <conf:(1)> lift:(1.25) lev:(0.1) [0] conv:(1)
2. d=false 4 ==> b=true 4      <conf:(1)> lift:(1.67) lev:(0.16) [1] conv:(1.6)
3. b=false 4 ==> d=true 4      <conf:(1)> lift:(1.67) lev:(0.16) [1] conv:(1.6)
4. a=true c=false 4 ==> e=true 4      <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.8)
5. a=true d=true 4 ==> e=true 4      <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.8)
6. c=false d=true 4 ==> e=true 4      <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.8)
7. a=true 7 ==> e=true 6      <conf:(0.86)> lift:(1.07) lev:(0.04) [0] conv:(0.7)
8. b=true 6 ==> e=true 5      <conf:(0.83)> lift:(1.04) lev:(0.02) [0] conv:(0.6)
9. d=true 6 ==> e=true 5      <conf:(0.83)> lift:(1.04) lev:(0.02) [0] conv:(0.6)
10. c=false 5 ==> a=true 4      <conf:(0.8)> lift:(1.14) lev:(0.05) [0] conv:(0.75)

2. Consider the data set and perform the Apriori Algorithm and FP algorithm support:3 and confidence=50%

Consider the market basket transactions shown in the above table.

(a) What is the maximum number of association rules that can be extracted

from this data (including rules that have zero support)?

(b) What is the maximum size of frequent itemsets that can be extracted

(assuming min sup > 0)?

Transaction ID	Items Bought
1	{Milk, Beer, Diapers}
2	{Bread, Butter, Milk}
3	{Milk, Diapers, Cookies}
4	{Bread, Butter, Cookies}
5	{Beer, Cookies, Diapers}
6	{Milk, Diapers, Bread, Butter}
7	{Bread, Butter, Diapers}
8	{Beer, Diapers}
9	{Milk, Diapers, Bread, Butter}
10	{Beer, Cookies}

INPUT:

@relation items

@attribute milk{t,f}

@attribute beer{t,f}

@attribute diapers{t,f}

@attribute bread{t,f}

@attribute butter{t,f}

@attribute cookies{t,f}

@data

t t t f f f

t f f t t f

t f t f f t

f f f t t t

f t t f f t

t f t t t f

f f t t t f

fttfff

tftttf

ftffff

## Output:

### APRIORI

```
Associate
Choose Apriori -N 10 -T 0.05 -D 0.05 -U 1.0 -M 0.3 -S 1.0 -c 1
Start Stop
Result list (right-click for ...
10:05:17 - Apriori
Associate output
=====
www
beer
diapers
bread
butter
cookies
==== Associate model (full training set) ===

Apriori
=====
Minimum support: 0.55 (5 instances)
Minimum metric <confidence>: 0.5
Number of cycles performed: 9

Generated sets of large itemsets:
Size of set of large itemsets L(1): 9
Size of set of large itemsets L(2): 5
Size of set of large itemsets L(3): 1

Best rules found:
1. bread=F 5 ==> beer=F 5   <conf:(1)> lift:(1.67) lev:(0.2) [2] convi:(2)
2. butter=F 5 ==> beer=F 5   <conf:(1)> lift:(1.67) lev:(0.2) [2] convi:(2)
3. butter=F 5 ==> bread=F 5   <conf:(1)> lift:(2) lev:(0.25) [2] convi:(2.5)
4. bread=F 5 ==> butter=F 5   <conf:(1)> lift:(2) lev:(0.25) [2] convi:(2.5)
5. beer=F 5 ==> bread=F 5   <conf:(1)> lift:(2) lev:(0.25) [2] convi:(2.5)
6. bread=F 5 ==> butter=F 5   <conf:(1)> lift:(2) lev:(0.25) [2] convi:(2.5)
7. bread=F butter=F 5 ==> beer=F 5   <conf:(1)> lift:(1.67) lev:(0.2) [2] convi:(2)
8. beer=F butter=F 5 ==> bread=F 5   <conf:(1)> lift:(2) lev:(0.25) [2] convi:(2.5)
9. beer=F bread=F 5 ==> butter=F 5   <conf:(1)> lift:(2) lev:(0.25) [2] convi:(2.5)
10. butter=F 5 ==> beer=F bread=F 5   <conf:(1)> lift:(2) lev:(0.25) [2] convi:(2.5)
```

### FP GROWTH

```
Associate
Choose FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.3
Start Stop
Result list (right-click for ...
10:05:17 - Apriori
10:06:24 - FPGrowth
Associate output
=====
Run information
=====
Scheme:      weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.3
Relation:    items
Instances:   10
Attributes:  6
milk
beer
diapers
bread
butter
cookies
==== Associate model (full training set) ===

FPGrowth found 4 rules (displaying top 4)
1. [butter=F]: 5 ==> [bread=F]: 5   <conf:(1)> lift:(2) lev:(0.25) convi:(2.5)
2. [bread=F]: 5 ==> [butter=F]: 5   <conf:(1)> lift:(2) lev:(0.25) convi:(2.5)
3. [milk=F, butter=F]: 3 ==> [bread=F]: 3   <conf:(1)> lift:(2) lev:(0.15) convi:(1.5)
4. [milk=F, bread=F]: 3 ==> [butter=F]: 3   <conf:(1)> lift:(2) lev:(0.15) convi:(1.5)
```

3.Bayes classification and decision tree (using training and test data)

<i>RID</i>	<i>age</i>	<i>income</i>	<i>student</i>	<i>credit_rating</i>	<i>Class: buys_computer</i>
1	<=30	high	no	fair	no
2	<=30	high	no	excellent	no
3	31 ... 40	high	no	fair	yes
4	>40	medium	no	fair	yes
5	>40	low	yes	fair	yes
6	>40	low	yes	excellent	no
7	31 ... 40	low	yes	excellent	yes
8	<=30	medium	no	fair	no
9	<=30	low	yes	fair	yes
10	>40	medium	yes	fair	yes
11	<=30	medium	yes	excellent	yes
12	31 ... 40	medium	no	excellent	yes
13	31 ... 40	high	yes	fair	yes
14	>40	medium	no	excellent	no

Input:

```
@relation decision_tree

@attribute age{young,middle,old}

@attribute income{low,medium,high}

@attribute student{yes,no}

@attribute Credit_rating{fair,excellent}

@attribute class{yes,no}

@data
```

young high no fair no

young high no excellent no

middle high no fair yes

old medium no fair yes

old low yes fair yes

old low yes excellent no

middle low yes excellent yes

young medium no fair no

young low yes fair yes

old medium yes fair yes

young medium yes excellent yes

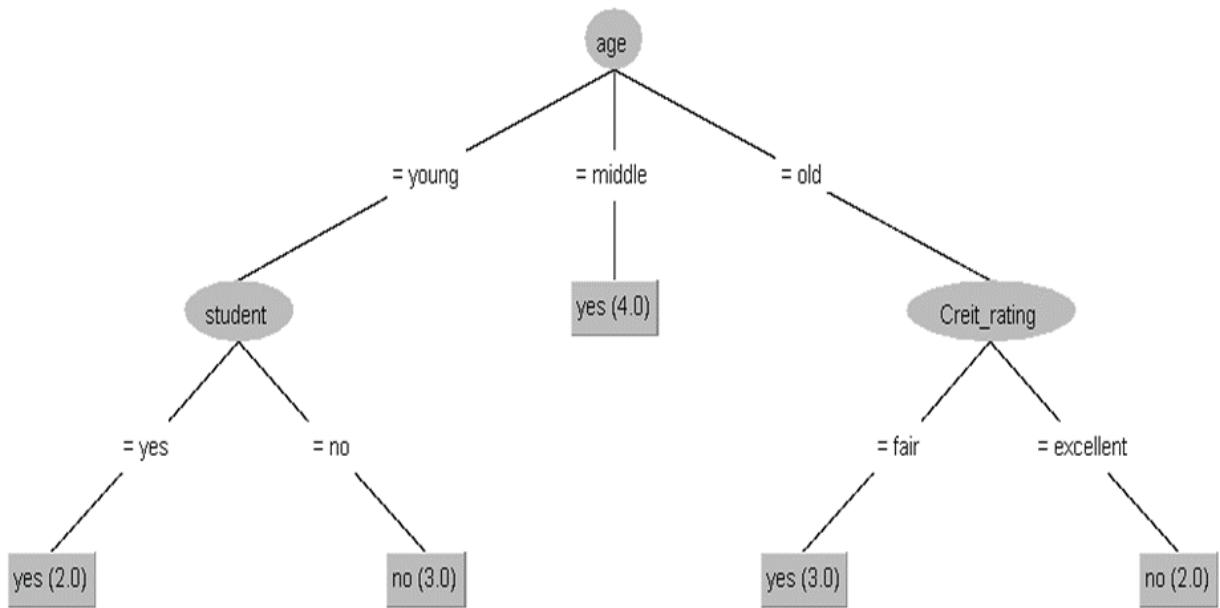
middle medium no excellent yes

middle high yes fair yes

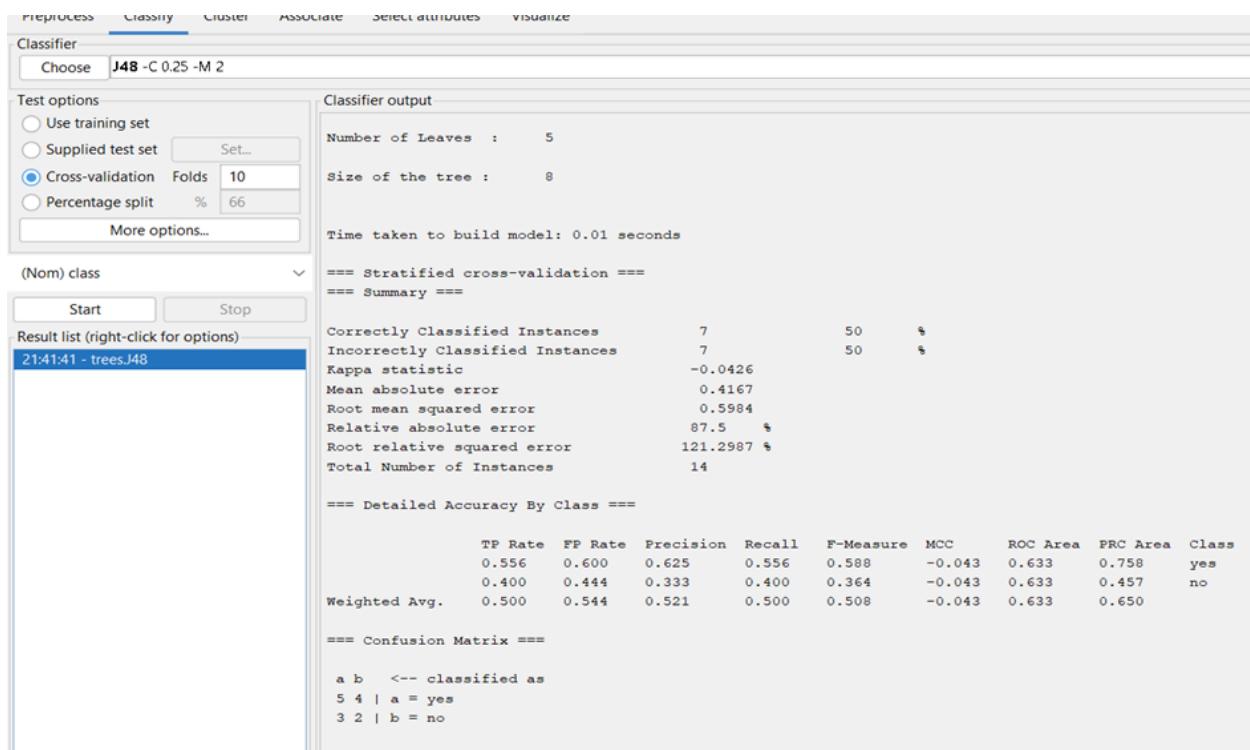
old medium no excellent no

output:

Tree:



## DECISION



4.Analysis the dataset “diabetes. csv” how the diabetes trend is for different age people, using linear regression and multiple regression.

Input:

```
data<-read.csv("C:/Users/Hari Naidu/Desktop/POM/download papers/diabetes.csv")
```

```
data
```

```
relation<-lm(data$Age~data$Outcome)
```

```
relation
```

```
relation<-lm(data$Age~data$Outcome+data$BMI)
```

```
relation
```

output:

```
--- [ reached 'max' / getoption("max.print") -- omitted 657 rows ] 
> relation<-lm(data$Age~data$Outcome+data$BMI)
> relation

call:
lm(formula = data$Age ~ data$Outcome + data$BMI)

Coefficients:
(Intercept)  data$Outcome      data$BMI
    32.84734       6.14177      -0.05469

> relation<-lm(data$Age~data$Outcome)
> relation

call:
lm(formula = data$Age ~ data$Outcome)

Coefficients:
(Intercept)  data$Outcome
    31.190       5.877
```

5.Implement using WEKA for the given Suppose a database has five transactions. Let min sup= 50%(2) and min con f= 80%.

<b>Transactions</b>	<b>Items</b>
T1	(M, O, N, K, E, Y)
T2	(D, O, N, K, E, Y)
T3	(M, A, K, E)
T4	(M, U, C, K, Y)
T5	(C, O, O, K, I, E)

- Find all frequent itemsets using Apriori algorithm

Also draw FP-Growth Tree

INPUT:

@relation items

@attribute M{t,f}

@attribute O{t,f}

@attribute N{t,f}

@attribute K{t,f}

@attribute E{t,f}

@attribute Y{t,f}

@attribute D{t,f}

@attribute A{t,f}

@attribute U{t,f}

@attribute C{t,f}

@attribute I{t,f}

@data

t t t t t f f f f f

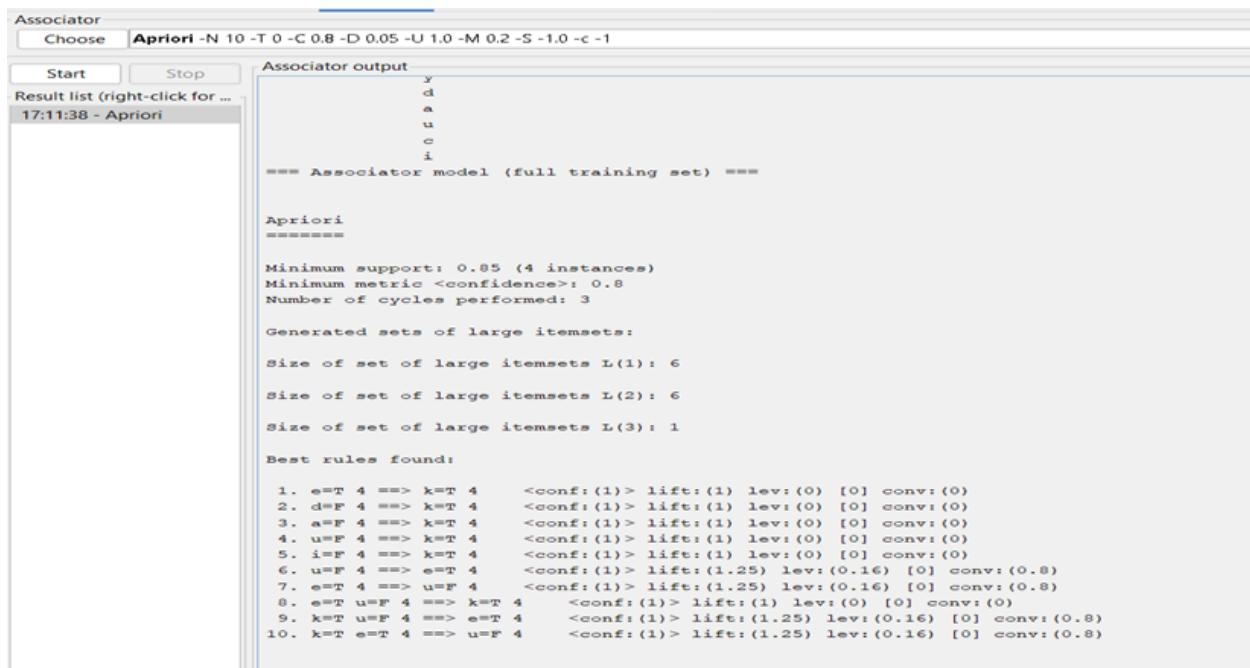
f t t t t t f f f f

t f f t t f f t f f f

t f f t f t f f t f f

f t f t t f f f f t t

Fp growth algorithm:



The screenshot shows the 'Associator' software interface with the 'Apriori' tab selected. The 'Result list' pane displays the command used: '17:11:38 - Apriori'. The 'Associator output' pane contains the following text:

```
Associator
Choose | Apriori -N 10 -T 0 -C 0.8 -D 0.05 -U 1.0 -M 0.2 -S -1.0 -c -1
Start | Stop
Result list (right-click for ...)

17:11:38 - Apriori

Associator output
y
d
a
u
c
i
==== Associator model (full training set) ====
Apriori
=====
Minimum support: 0.85 (4 instances)
Minimum metric <confidence>: 0.8
Number of cycles performed: 3
Generated sets of large itemsets:
Size of set of large itemsets L(1): 6
Size of set of large itemsets L(2): 6
Size of set of large itemsets L(3): 1
Best rules found:
1. e=T 4 ==> k=T 4      <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
2. d=F 4 ==> k=T 4      <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
3. a=F 4 ==> k=T 4      <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
4. u=F 4 ==> k=T 4      <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
5. i=F 4 ==> k=T 4      <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
6. u=F 4 ==> e=T 4      <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
7. e=T 4 ==> u=F 4      <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
8. e=T u=F 4 ==> k=T 4  <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
9. k=T u=F 4 ==> e=T 4  <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
10. k=T e=T 4 ==> u=F 4 <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
```

Fp growth algorithm:

Associator

Choose FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.5

Start Stop

Result list (right-click for ...)

17:11:38 - Apriori  
17:13:15 - FPGrowth

Associator output

```
==== Run information ====
Scheme:      weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.5
Relation:    database
Instances:   5
Attributes:  11
m
o
n
k
e
y
d
a
u
c
i
==== Associator model (full training set) ====
FPGrowth found 7 rules (displaying top 7)

1. [c=F]: 3 ==> [u=F]: 3  <conf:(1)> lift:(1.25) lev:(0.12) conv:(0.6)
2. [c=F]: 3 ==> [i=F]: 3  <conf:(1)> lift:(1.25) lev:(0.12) conv:(0.6)
3. [n=F]: 3 ==> [d=F]: 3  <conf:(1)> lift:(1.25) lev:(0.12) conv:(0.6)
4. [u=F, i=F]: 3 ==> [c=F]: 3  <conf:(1)> lift:(1.67) lev:(0.24) conv:(1.2)
5. [c=F]: 3 ==> [u=F, i=F]: 3  <conf:(1)> lift:(1.67) lev:(0.24) conv:(1.2)
6. [u=F, c=F]: 3 ==> [i=F]: 3  <conf:(1)> lift:(1.25) lev:(0.12) conv:(0.6)
7. [i=F, c=F]: 3 ==> [u=F]: 3  <conf:(1)> lift:(1.25) lev:(0.12) conv:(0.6)
```

6. Prediction of Categorical Data using Decision Tree Algorithm through WEKA using any datasets. a) Tree b) Preprocess c) Logistic

Output:

Tree:

Preprocess Classify Cluster Associate Select attributes Visualize

**Classifier**

Choose J48 -C 0.25 -M 2

**Test options**

- Use training set
- Supplied test set Set...
- Cross-validation Folds 10
- Percentage split % 66

[More options...](#)

**(Nom) class**

[Start](#) [Stop](#)

**Result list (right-click for options)**

- 12:51:54 - bayes.NaiveBayes
- 12:58:01 - bayes.NaiveBayes
- 13:06:38 - trees.J48
- 13:09:14 - functions.Logistic
- 13:10:16 - functions.Logistic
- 13:11:08 - trees.J48

**Classifier output**

```

Number of Leaves : 20
Size of the tree : 39
Time taken to build model: 0.01 seconds

*** Stratified cross-validation ***
*** Summary ***

Correctly Classified Instances      567      73.8281 %
Incorrectly Classified Instances   201      26.1719 %
Kappa statistic                   0.4164
Mean absolute error               0.3158
Root mean squared error          0.4463
Relative absolute error           69.4841 %
Root relative squared error      93.6293 %
Total Number of Instances        768

*** Detailed Accuracy By Class ***

      TP Rate  FP Rate  Precision  Recall  F-Measure  MCC  ROC Area  PRC Area  Class
          0.814    0.403     0.790    0.814     0.802     0.417    0.751    0.811  tested_negative
          0.597    0.186     0.632    0.597     0.614     0.417    0.751    0.572  tested_positive

Weighted Avg.      0.738    0.327     0.735    0.738     0.736     0.417    0.751    0.727

*** Confusion Matrix ***

      a   b  <-- classified as
  407  93 |  a = tested_negative
  108 160 |  b = tested_positive

```

## Preprocessor:

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

[Open file...](#) [Open URL...](#) [Open DB...](#) [Generate...](#) [Undo](#) [Edit...](#) [Save...](#)

**Filter**

Choose **None**

**Current relation**

Relation: pima\_diabetes Attributes: 9 Sum of weights: 768

**Attributes**

No.	Name
1	preg
2	plas
3	pres
4	skin
5	insu
6	mass
7	pedi
8	age
9	class

[Remove](#)

**Selected attribute**

Name: preg  
Missing: 0 (0%)  
Distinct: 17  
Type: Numeric  
Unique: 2 (0%)

Statistic	Value
Minimum	0
Maximum	17
Mean	3.845
StdDev	3.37

**Class: class (Nom)** [Visualize All](#)

Class	Value	Count
0	0	246
1	0	105
0	1	76
1	1	125
0	2	50
1	2	45
0	3	66
1	3	24
0	4	11
1	4	19
0	5	2
1	5	1
0	6	1
1	6	1

## logistic:

The screenshot shows the Weka interface for the Logistic classifier. The 'Classify' tab is selected. Under 'Classifier', 'Logistic' is chosen with parameters: -R 1.0E-8 -M -1 -num-decimal-places 4. The 'Test options' section shows 'Cross-validation' selected with 10 folds. The 'Classifier output' pane displays classification statistics for three attributes: mass (0.9142), pedi (0.3886), and age (0.9852). It also shows the time taken to build the model (0.01 seconds) and stratified cross-validation details. The 'Result list' pane shows a history of runs, with the current one being '13:09:14 - functions.Logistic'. The 'Detailed Accuracy By Class' table provides accuracy metrics for two classes: tested\_neg and tested\_pos. The 'Confusion Matrix' section shows the counts for correctly classified instances (440, 60) and misclassified instances (115, 153).

Class	TP	FP	FN	TN
tested_neg	0.880	0.429	0.120	0.571
tested_pos	0.772	0.321	0.718	0.228
<b>Avg.</b>	0.772	0.321	0.767	0.228

7.Create the dataset using ARFF file format:

Transaction ID	Items
T1	Hot Dogs, Buns, Ketchup
T2	Hot Dogs, Buns
T3	Hot Dogs, Coke, Chips
T4	Chips, Coke
T5	Chips, Ketchup
T6	Hot Dogs, Coke, Chips

a.Find the **frequent itemsets** and generate **association rules** on this. Assume that minimum support threshold ( $s = 33.33\%$ ) and minimum confident threshold ( $c = 60\%$ ).

b.List the various rule generated by apriori and FP tree algorithm ,mention whether accepted or rejected.

**Input:**

@relation hotdogs

@attribute hotdogs {t,f}

@attribute buns {t,f}

@attribute ketchup {t,f}

@attribute coke {t,f}

@attribute chips {t,f}

@data

t t t f f

t t f f f

t f f t t

f f f t t

f f t f t

t f f t t

**output:**

apriori algorithm:

Associator  
Choose **Apriori** -N 10 -T 0 -C 0.6 -D 0.05 -U 1.0 -M 0.2 -S -1.0 -c -1

Start Stop

Result list (right-click for ...)

12:22:41 - Apriori  
12:23:31 - FP Growth  
12:23:56 - FP Growth  
12:27:32 - FP Growth  
12:27:34 - FP Growth  
13:23:39 - Apriori

Associator output  
==== Associator model (full training set) ===

Apriori  
=====

Minimum support: 0.55 (3 instances)  
Minimum metric <confidence>: 0.6  
Number of cycles performed: 9

Generated sets of large itemsets:

Size of set of large itemsets L(1): 6  
Size of set of large itemsets L(2): 7  
Size of set of large itemsets L(3): 4  
Size of set of large itemsets L(4): 1

Best rules found:

1. chips=t 4 ==> buns=f 4 <conf:(1)> lift:(1.5) lev:(0.22) [1] conv:(1.33)
2. buns=f 4 ==> chips=t 4 <conf:(1)> lift:(1.5) lev:(0.22) [1] conv:(1.33)
3. coke=t 3 ==> buns=f 3 <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)
4. coke=t 3 ==> ketchup=f 3 <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)
5. coke=t 3 ==> chips=t 3 <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)
6. ketchup=f coke=t 3 ==> buns=f 3 <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)
7. buns=f coke=t 3 ==> ketchup=f 3 <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)
8. buns=f ketchup=f 3 ==> coke=t 3 <conf:(1)> lift:(2) lev:(0.25) [1] conv:(1.5)
9. coke=t 3 ==> buns=f ketchup=f 3 <conf:(1)> lift:(2) lev:(0.25) [1] conv:(1.5)
10. ketchup=f chips=t 3 ==> buns=f 3 <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)

Fp growth:

The screenshot shows the WEKA Associate interface with the following details:

- Preprocess**, **Classify**, **Cluster**, **Associate** (selected), **Select attributes**, **Visualize**
- Associator** tab selected.
- Choose**: **FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.2**
- Start** and **Stop** buttons.
- Associator output** pane:
 

```
== Run information ==
Scheme: weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.2
Relation: hotdogs
Instances: 6
Attributes: 5
hotdogs
buns
ketchup
coke
chips
== Associator model (full training set) ==
FPGrowth found 2 rules (displaying top 2)
1. [hotdogs=f]: 2 ==> [buns=f]: 2 <conf:(1)> lift:(1.5) lev:(0.11) conv:(0.67)
2. [chips=f]: 2 ==> [coke=f]: 2 <conf:(1)> lift:(2) lev:(0.17) conv:(1)
```
- Result list** (right-click for ...):
  - 12:22:41 - Apriori
  - 12:23:31 - FPGrowth
  - 12:23:56 - FPGrowth
  - 12:27:32 - FPGrowth
  - 12:27:34 - FPGrowth
  - 13:23:39 - Apriori
  - 13:24:49 - FPGrowth

8.Prediction of Categorical Data using Rule base classification and decision tree classification through WEKA using any datasets. Compare the accuracy using two algorithm and plot the graph

#### Day 4 experiments:

1.1.Consider that you are owning a supermarket mall and through membership cards, you have some basic data about your customers like Customer ID, age, gender, annual income and spending score. For the above scenario, the Problem Statement was You want to understand the customers who can easily converge [Target Customers] so that the data can be given to the marketing team and plan the strategy accordingly. For the above scenario prepare a dataset and perform **Clustering Analysis** to segment the customers in the Mall. There are clearly Five segments of Customers based on their Annual Income and Spending Score namely *Usual*

*Customers, Priority Customers, Senior Citizen Target Customers, and Young Target Customers.* Sample data

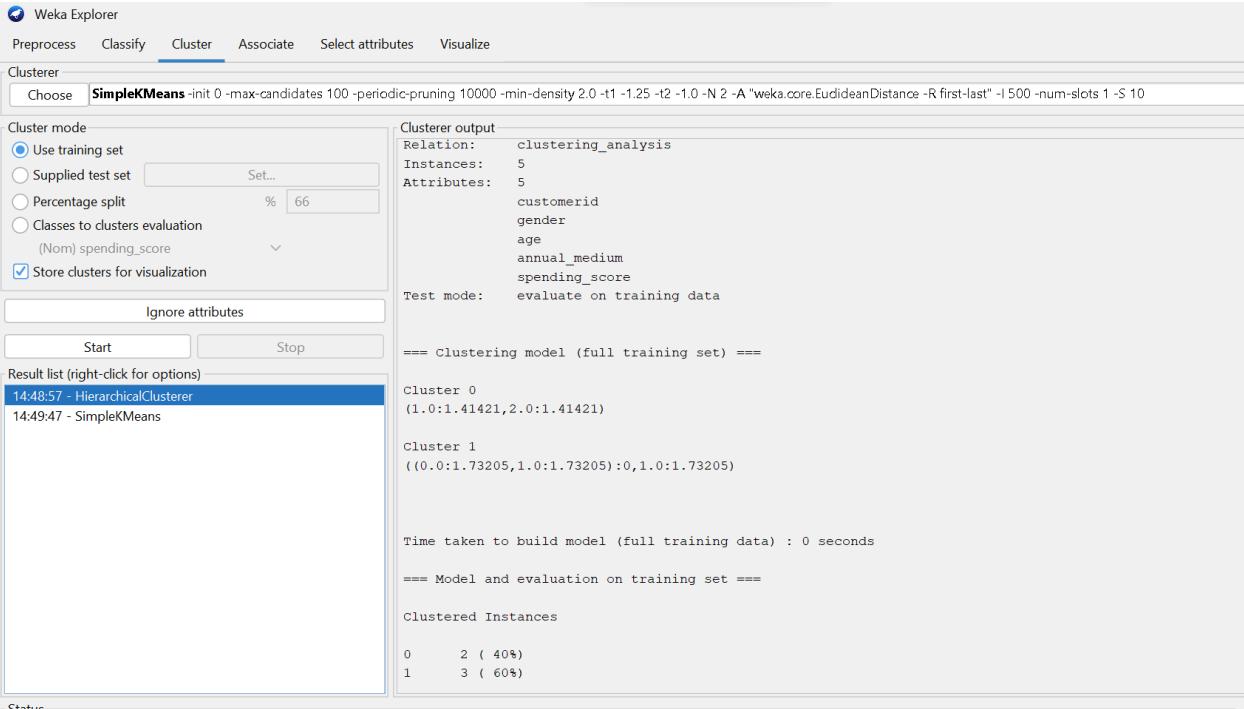
	<b>CustomerID</b>	<b>Gender</b>	<b>Age</b>	<b>Annual Income (k\$)</b>	<b>Spending Score (1-100)</b>
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

INPUT:

```
@relation clustering_analysis
@attribute customerid{1,2,3,4,5}
@attribute gender{male,female}
@attribute age{young,middle,old}
@attribute annual_medium{low,half,high}
@attribute spending_score{least,medium,highest}
@data
1 male young low medium
2 male young low highest
3 female young half least
4 female middle half medium
5 female old high medium
```

## OUTPUT:

### HIERARCHICAL CLUSTERING



The screenshot shows the Weka Explorer interface with the 'Cluster' tab selected. The 'Choose' dropdown is set to 'SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -l 500 -num-slots 1 -S 10'. The 'Cluster mode' section has 'Use training set' selected. The 'Clusterer output' pane displays the clustering analysis results, including the relation name 'clustering\_analysis', number of instances (5), attributes (customerid, gender, age, annual\_medium, spending\_score), and test mode ('evaluate on training data'). The results show two clusters: Cluster 0 (1.0:1.41421, 2.0:1.41421) and Cluster 1 ((0.0:1.73205, 1.0:1.73205):0, 1.0:1.73205). The time taken to build the model was 0 seconds. The final clustered instances are 0 (2 (40%)) and 1 (3 (60%)).

```

Choose SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -l 500 -num-slots 1 -S 10

Cluster mode
 Use training set
 Supplied test set Set...
 Percentage split % 66
 Classes to clusters evaluation
(Nom) spending_score
 Store clusters for visualization

Clusterer output
Relation: clustering_analysis
Instances: 5
Attributes: 5
customerid
gender
age
annual_medium
spending_score
Test mode: evaluate on training data

==== Clustering model (full training set) ====
Cluster 0
(1.0:1.41421, 2.0:1.41421)

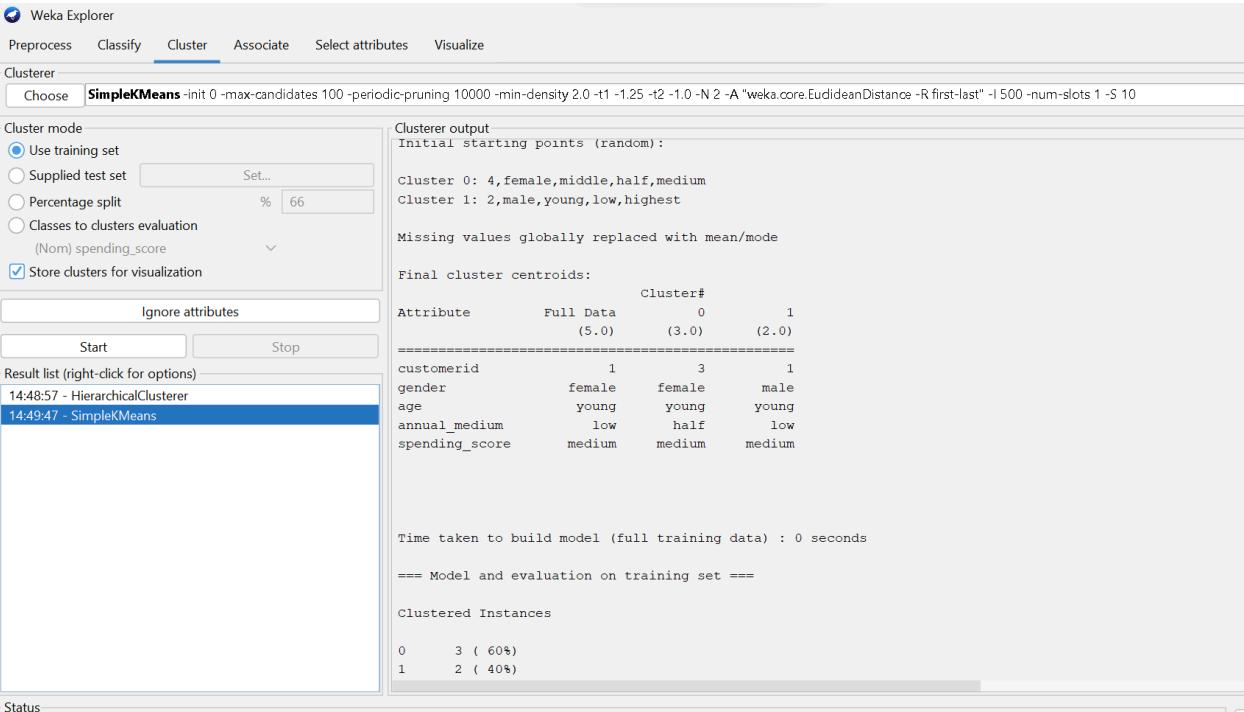
Cluster 1
((0.0:1.73205, 1.0:1.73205):0, 1.0:1.73205)

Time taken to build model (full training data) : 0 seconds
==== Model and evaluation on training set ====
Clustered Instances

0      2 ( 40%)
1      3 ( 60%)

```

### K MEANS CLUSTERING



The screenshot shows the Weka Explorer interface with the 'Cluster' tab selected. The 'Choose' dropdown is set to 'SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -l 500 -num-slots 1 -S 10'. The 'Cluster mode' section has 'Use training set' selected. The 'Clusterer output' pane displays the clustering analysis results, including initial starting points (random), cluster centroids (Cluster 0: 4, female, middle, half, medium; Cluster 1: 2, male, young, low, highest), and final cluster centroids for attributes customerid, gender, age, annual\_medium, and spending\_score. The time taken to build the model was 0 seconds. The final clustered instances are 0 (3 (60%)) and 1 (2 (40%)).

```

Choose SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -l 500 -num-slots 1 -S 10

Cluster mode
 Use training set
 Supplied test set Set...
 Percentage split % 66
 Classes to clusters evaluation
(Nom) spending_score
 Store clusters for visualization

Clusterer output
Initial starting points (random):
Cluster 0: 4, female, middle, half, medium
Cluster 1: 2, male, young, low, highest

Missing values globally replaced with mean/mode

Final cluster centroids:
Attribute          Full Data   Cluster#
                  (5.0)       0           1
=====
customerid        1           3           1
gender            female     female     male
age               young      young      young
annual_medium    low        half       low
spending_score   medium     medium     medium

Time taken to build model (full training data) : 0 seconds
==== Model and evaluation on training set ====
Clustered Instances

0      3 ( 60%)
1      2 ( 40%)

```

