G P Abinaya

192211241

CSA1679: DATA WAREHOUSING AND DATA MINING FOR INDUSTRY

DAY 1:

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

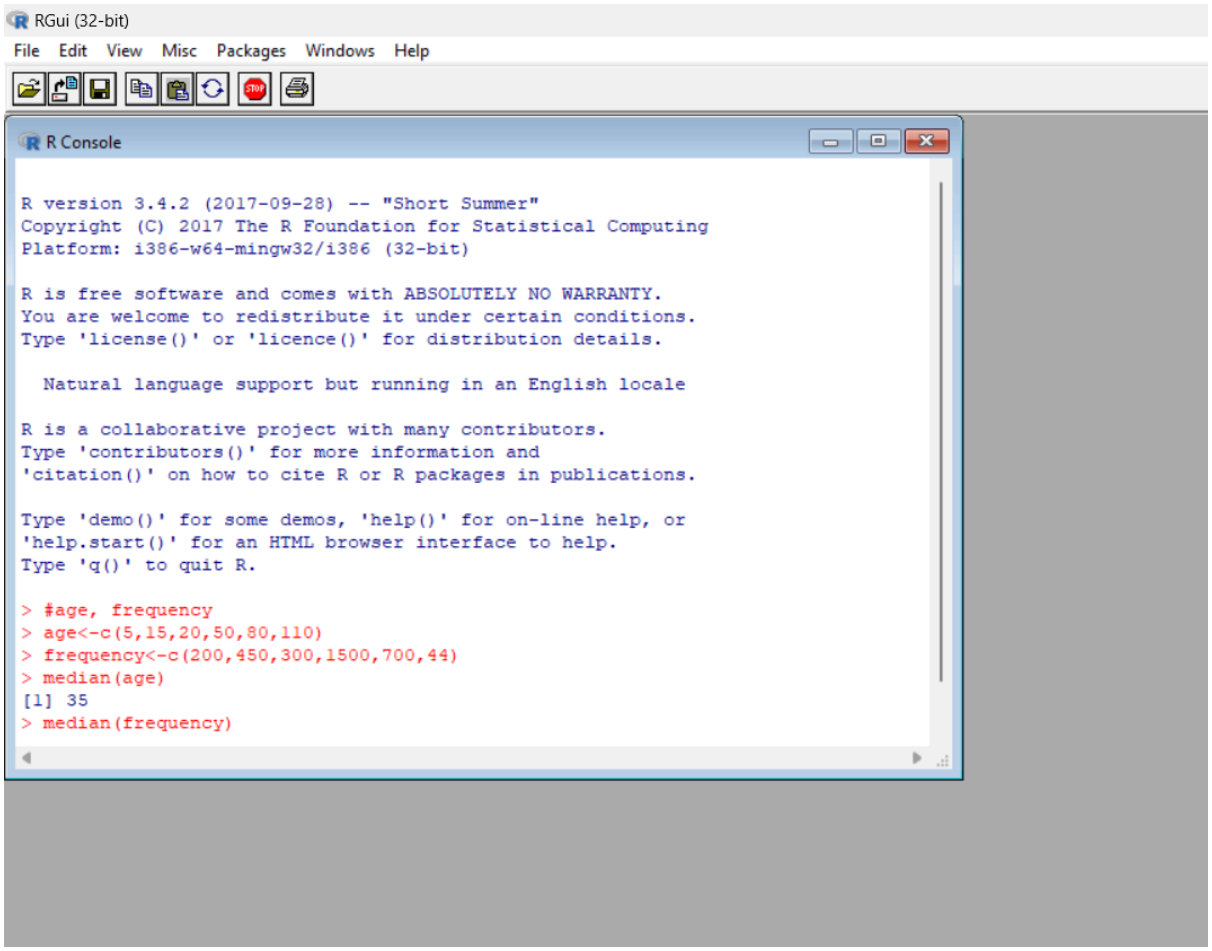
80-110 44

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:



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.).

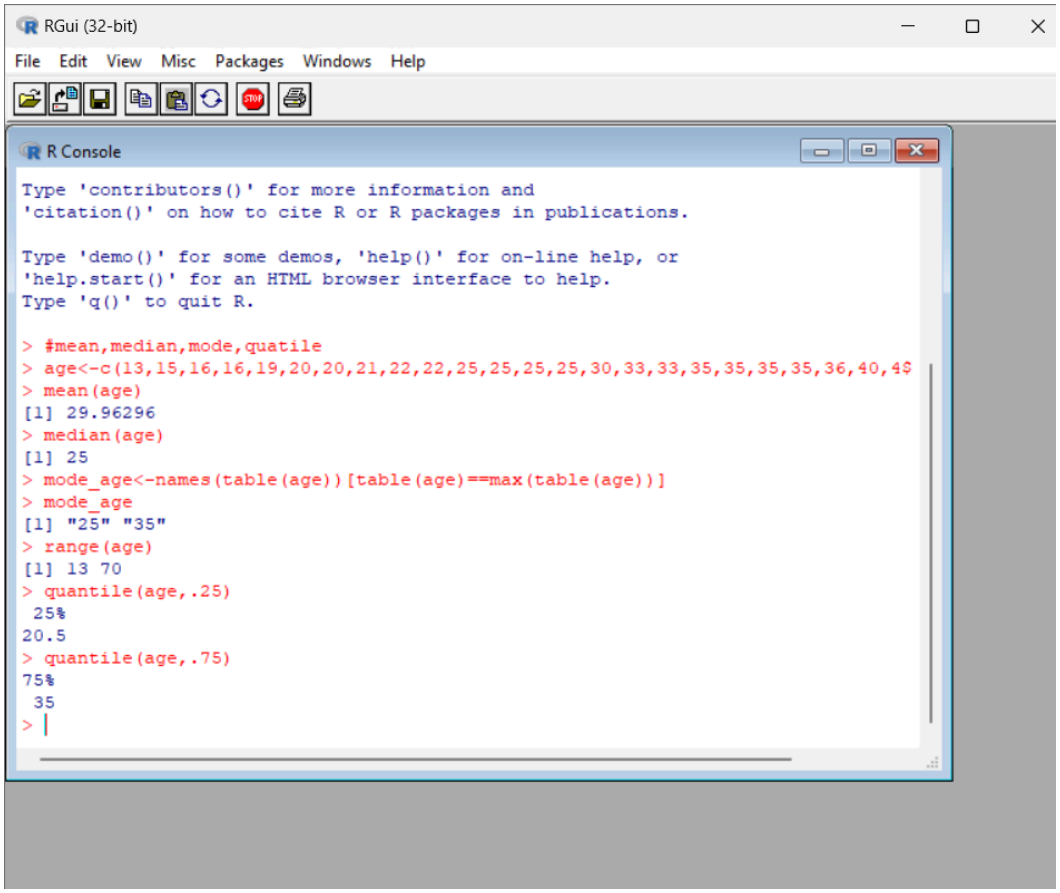
(c) What is the midrange of the data?

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

Input:

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



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

OUTPUT:

```
> 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
(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

> q
function (save = "default", status = 0, runLast = TRUE)
.Internal(quit(save, status, runLast))
<bytecode: 0x000002913c309348>
<environment: namespace:base>
```

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

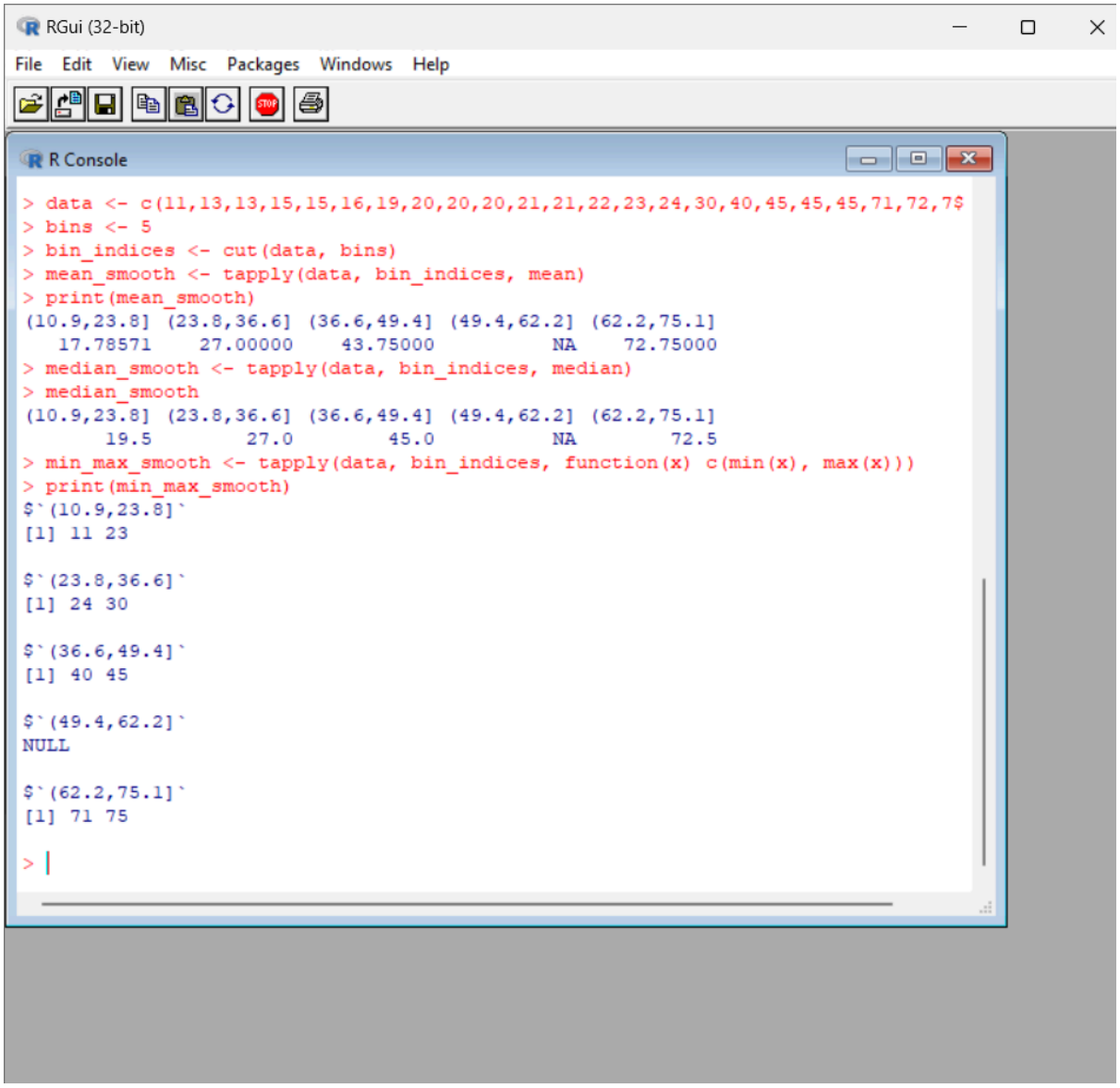
b) Smoothing by bin median

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:



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

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)

```
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:

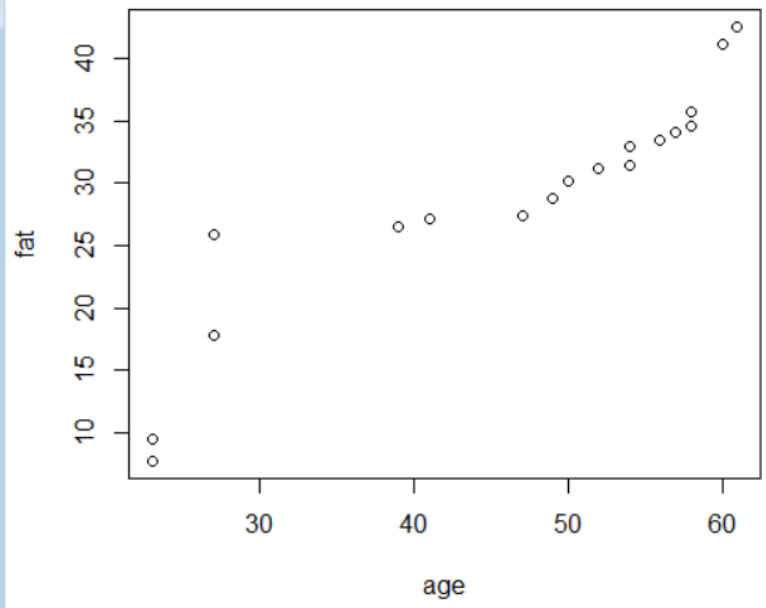
RGui (32-bit)

File History Resize Windows

R Console

```
[1] 40 45

$`(49.4,62.2]`
NULL

$`(62.2,75.1]`
[1] 71 75

> 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$
> 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)
>
```

R Graphics: Device 2 (ACTIVE)



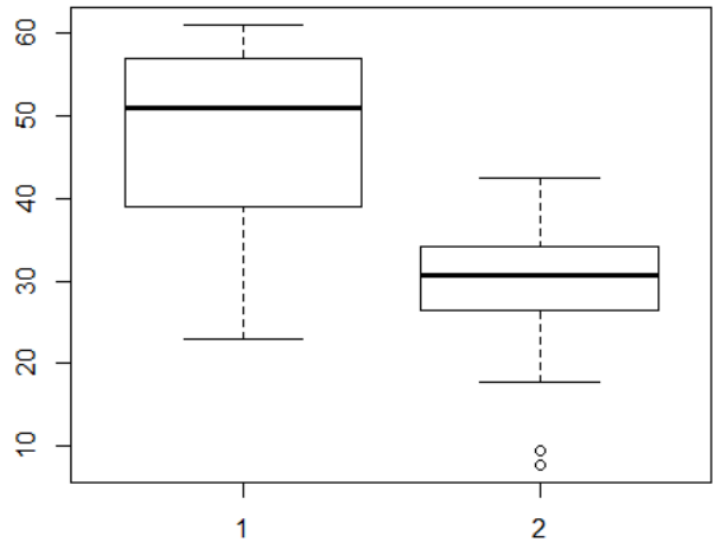
RGui (32-bit)

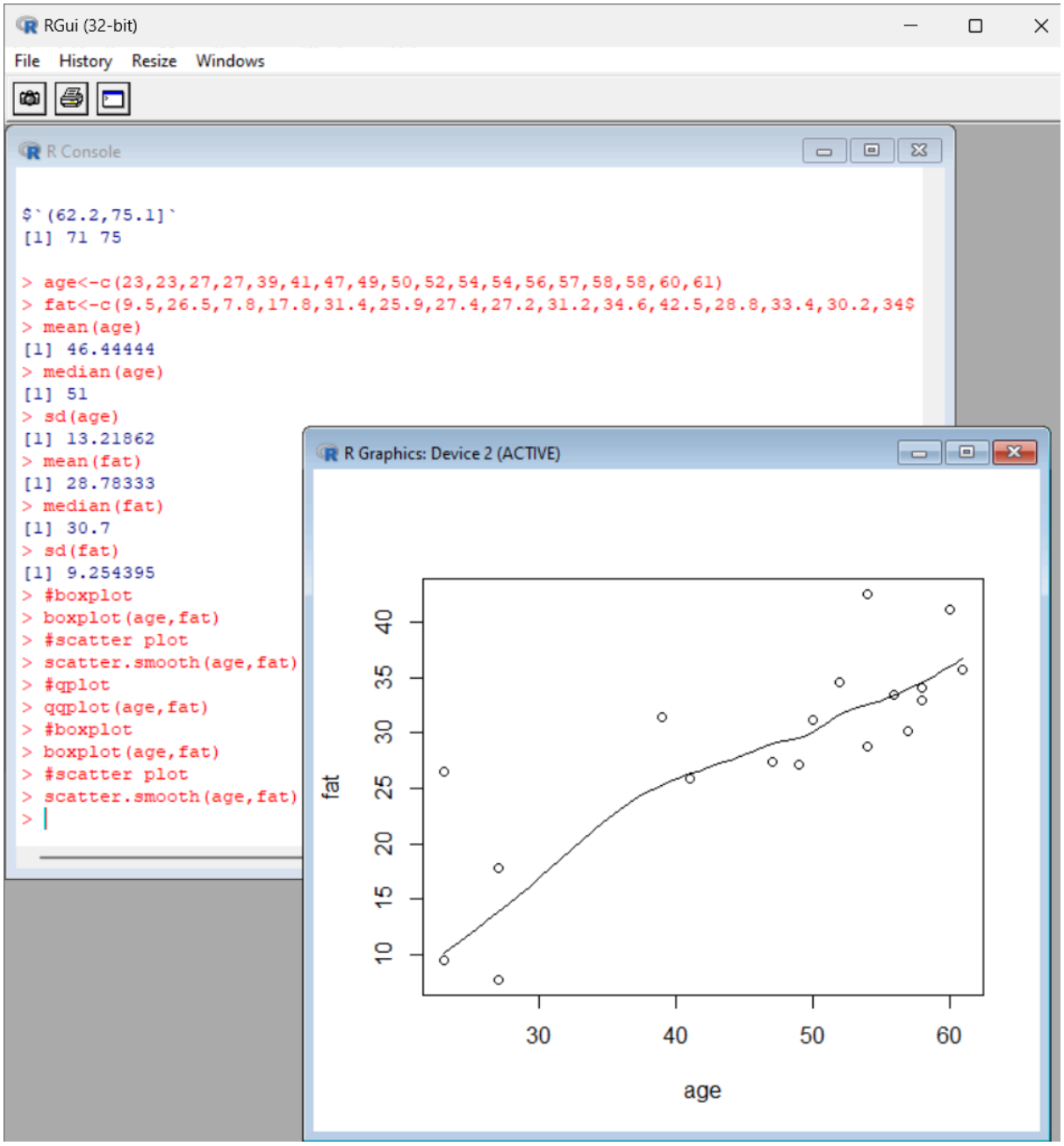
File History Resize Windows

R Console

```
$`(49.4,62.2]`
NULL

$`(62.2,75.1]`
[1] 71 75

> 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$
> 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)
> #boxplot
> boxplot(age,fat)
>
```

R Graphics: Device 2 (ACTIVE)





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].

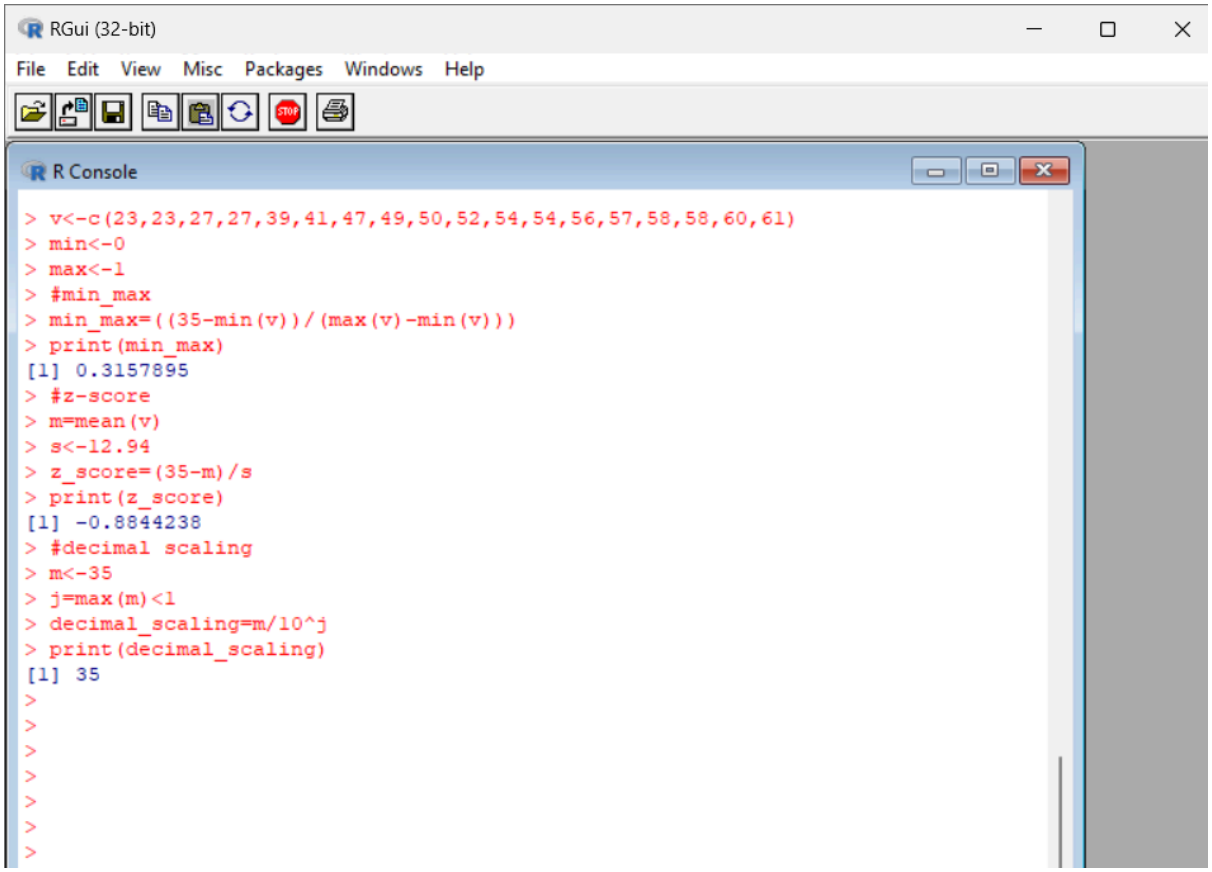
(ii) Use z-score normalization to transform the value 35 for age, where the standard deviation of age is 12.94 years.

(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:



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)

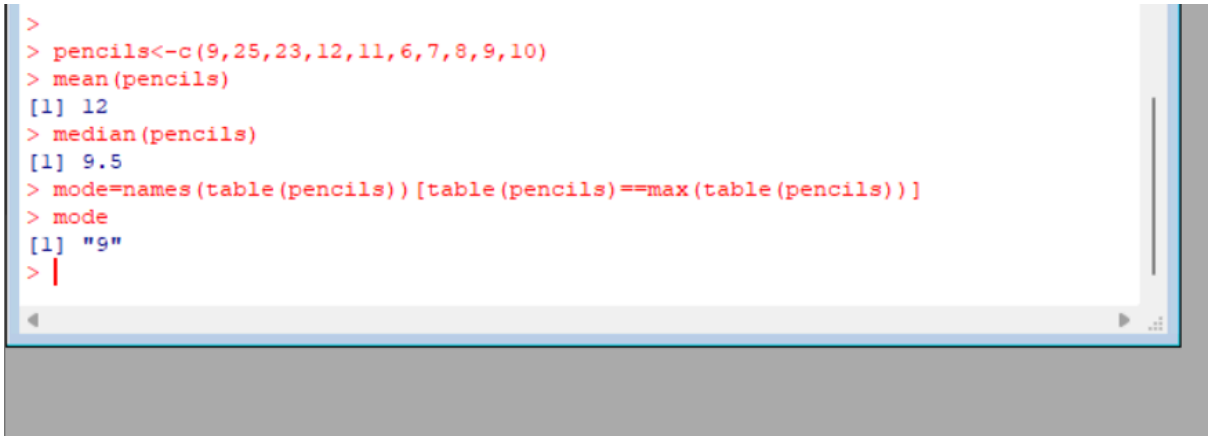
mean(pencils)

median(pencils)

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

mode

Output:



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:

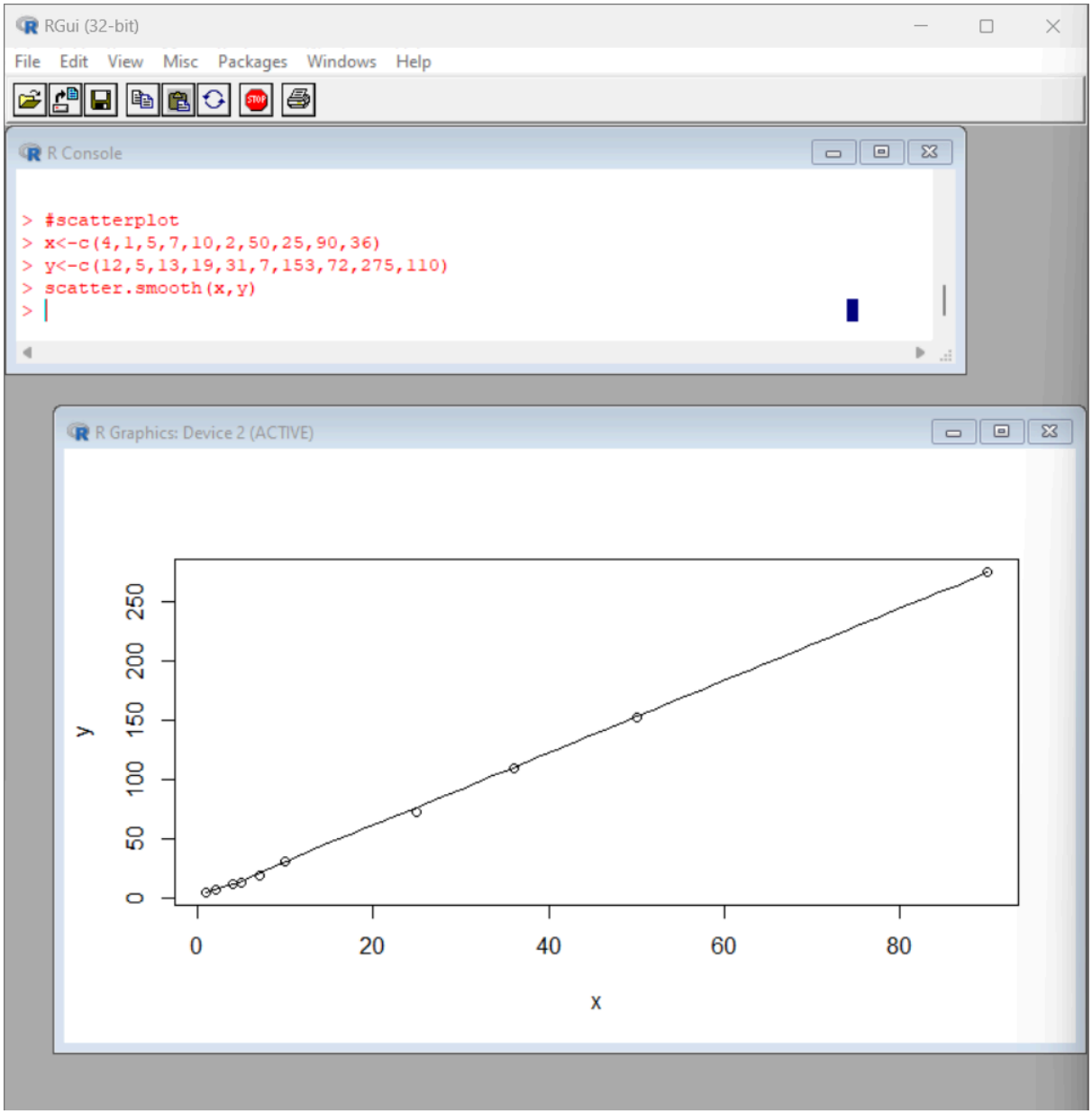
#scatterplot

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:

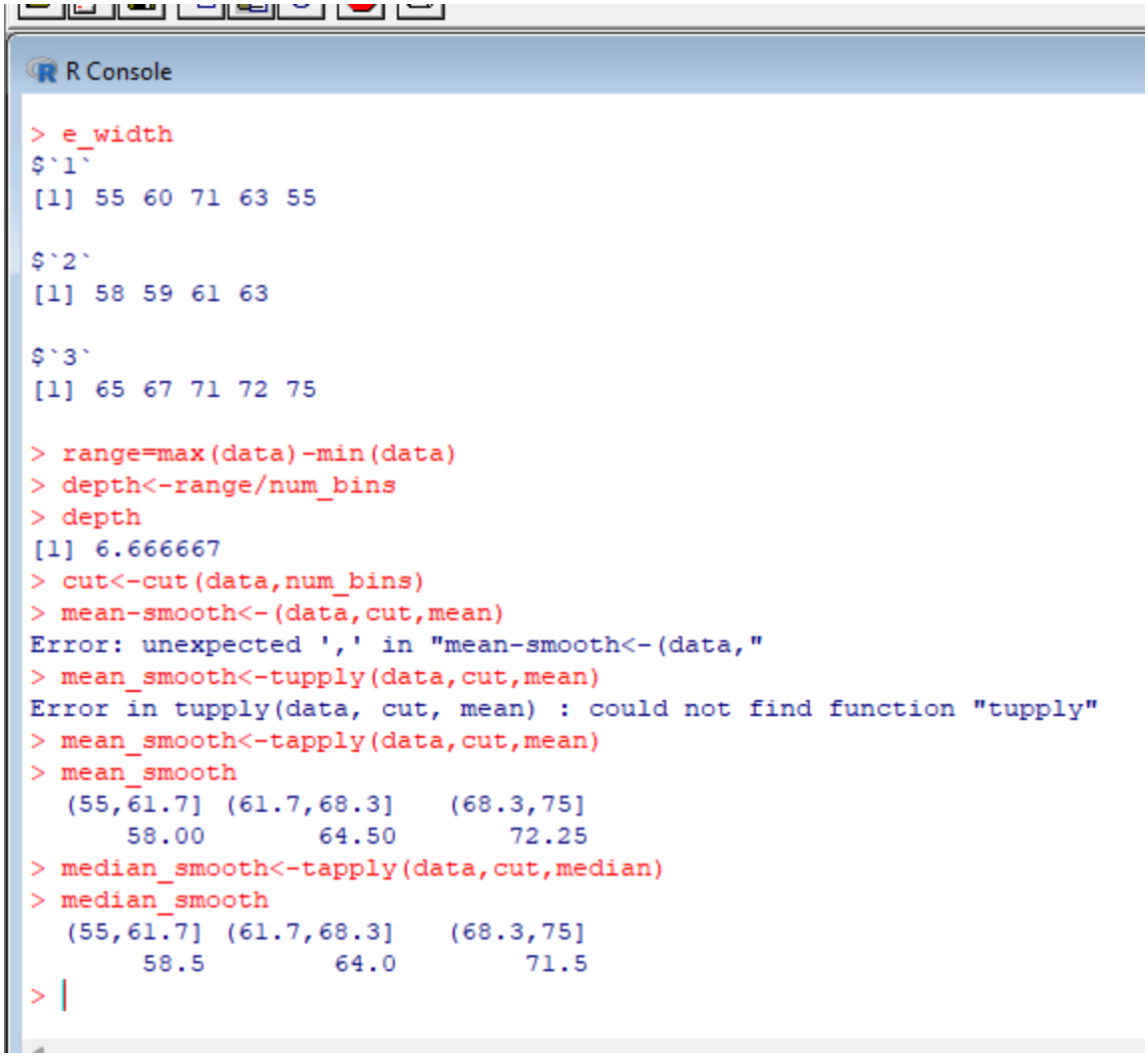
marks <- c(55, 60, 71, 63, 55, 65, 50, 55, 58, 59, 61, 63, 65, 67, 71, 72, 75)

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:

```
  10.1927
1] 10.1927
 data<-78,81,82,74,83,84,82,77,80,70)
rror: unexpected ',' in "data<-78,"
 data<-c(78,81,82,74,83,84,82,77,80,70)
 IQR(data)
1] 4.75
```

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

quantile(age,.25)

quantile(age,.75)

output:

```
[1] 4.75
> 11.1927
[1] 11.1927
> age<-c(13,15,16,16,19,20,20,21,22,22,25,25,25,25,30,33,33,35,35,35,35,36,40,4$
> quantile(age,.25)
 25%
20.5
> quantile(age,.75)
75%
 35
>
```

DAY2:

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

Use cov() to calculate the sample covariance between B and C.

Use another call to cov() to calculate the sample covariance matrix for the preferences.

Use cor() to calculate the sample correlation between B and C.

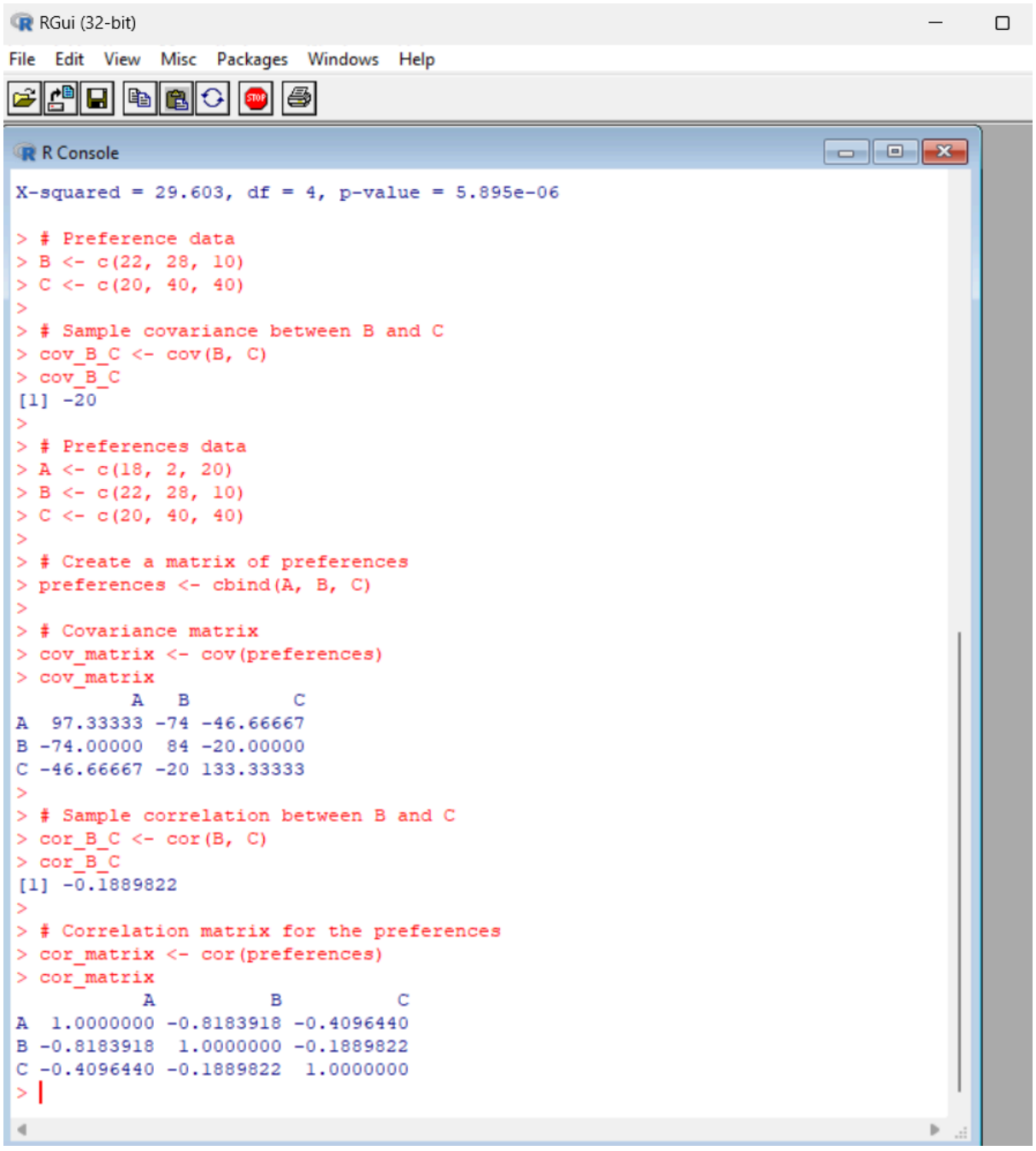
Use another call to cor() to calculate the sample correlation matrix for the preferences.

18, 18, 18, 20, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 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

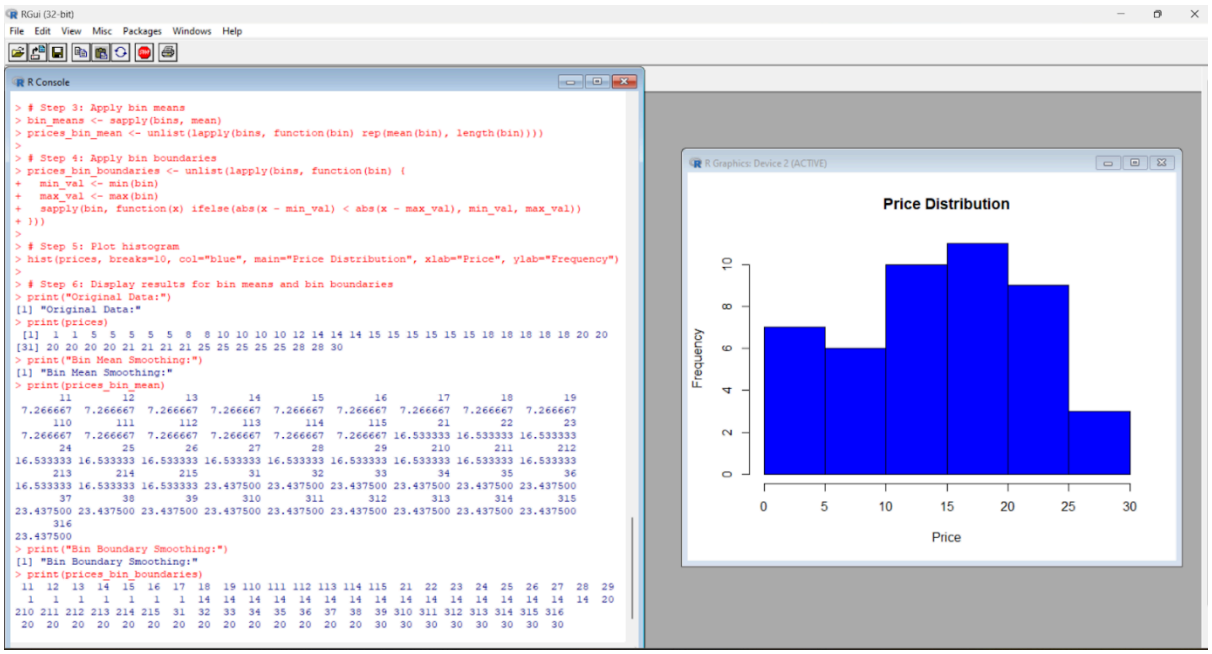


2 question:

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, 18, 18, 18, 18, 18,

(i)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



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, 8476,35,47,64,95,66,89,36,84

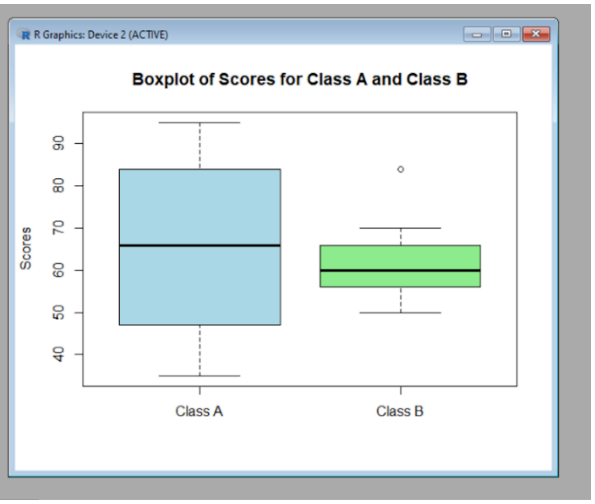
Class B: 51, 56, 84, 60, 59, 70, 63, 66, 5051,56,84,60,59,70,63,66,50

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

(ii) Plot above in boxplot and give the inferences

Class B: 51, 56, 84, 60, 59, 70, 63, 66, 5051,56,84,60,59,70,63,66,50

```
> # Input data
> class_A <- c(76, 35, 47, 64, 95, 66, 89, 36, 84)
> class_B <- c(51, 56, 84, 60, 59, 70, 63, 66, 50)
> # Mean
> mean_A <- mean(class_A)
> mean_B <- mean(class_B)
>
> # Median
> median_A <- median(class_A)
> median_B <- median(class_B)
>
> # Range (Max - Min)
> range_A <- range(class_A)
> range_B <- range(class_B)
> range_diff_A <- diff(range_A)
> range_diff_B <- diff(range_B)
>
> # Print the results
> cat("Class A - Mean:", mean_A, " Median:", median_A, " Range:", range_diff_A,$
Class A - Mean: 65.77778  Median: 66  Range: 60
> cat("Class B - Mean:", mean_B, " Median:", median_B, " Range:", range_diff_B,$
Class B - Mean: 62.11111  Median: 60  Range: 34
> # Boxplot
> boxplot(class_A, class_B,
+         names = c("Class A", "Class B"),
+         main = "Boxplot of Scores for Class A and Class B",
+         ylab = "Scores",
+         col = c("lightblue", "lightgreen"))
>
> # Adding grid lines for better readability
> grid()
```

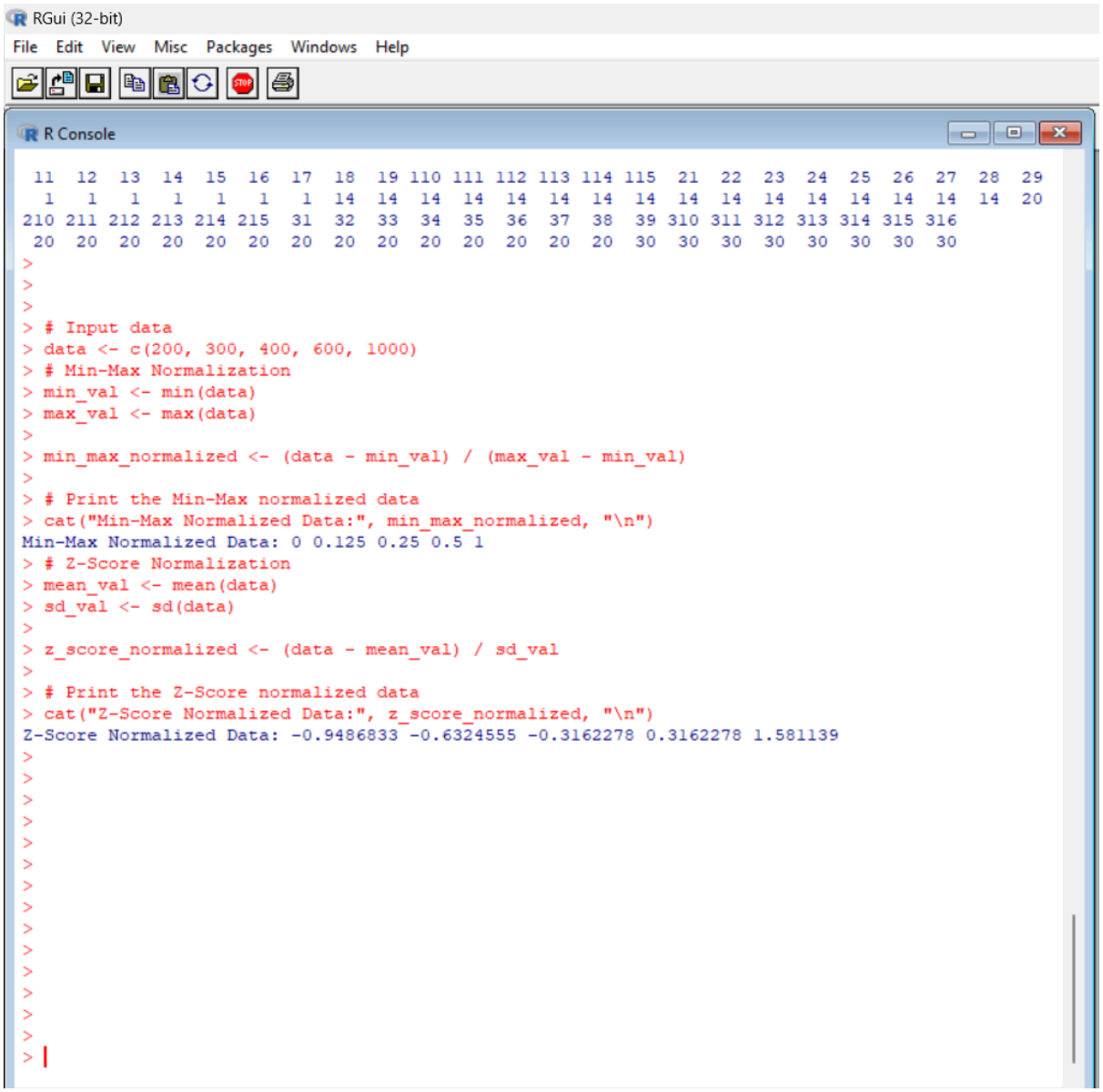


4. Let us consider one example to make the calculation method clear. Assume that the minimum and maximum values for the feature F are $50,000 and $100,000 correspondingly. It needs to range F from 0 to 1. In accordance with min-max normalization, v = $80,

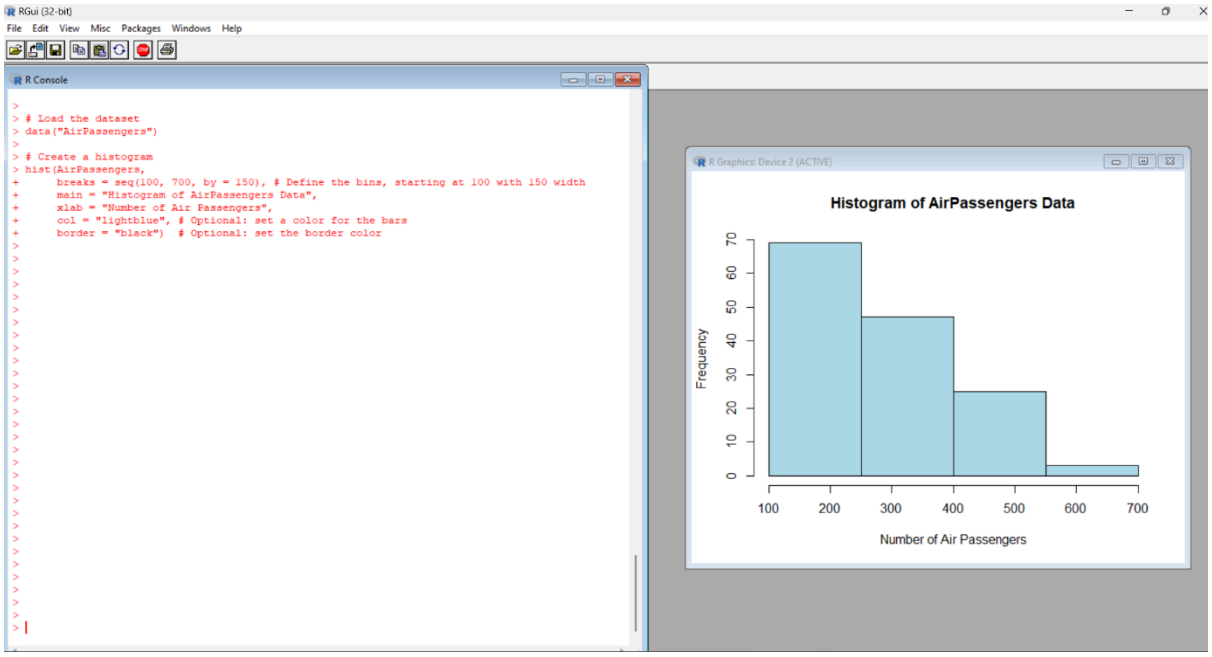
b) Use the two methods below to normalize the following group of data: 200, 300, 400, 600, 1000

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

(b) z-score normalization



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

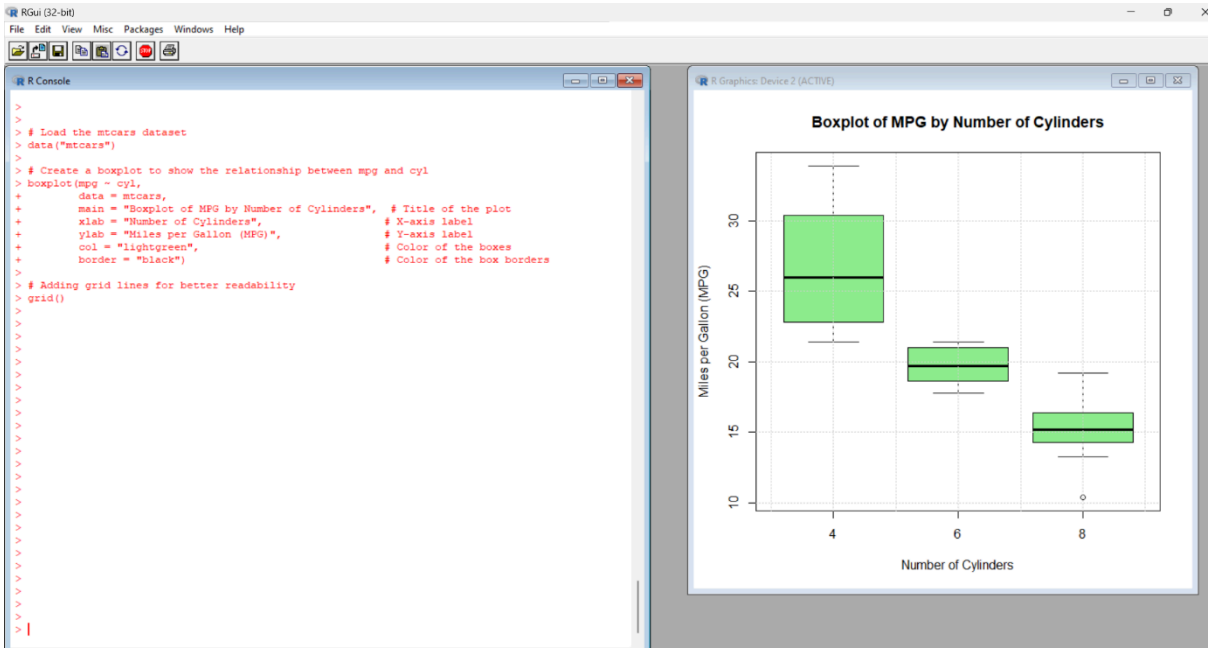


6. Obtain Multiple Lines in Line Chart using a single Plot Function in R.Use attributes“mpg”and“qsec”of the dataset “mtcars”

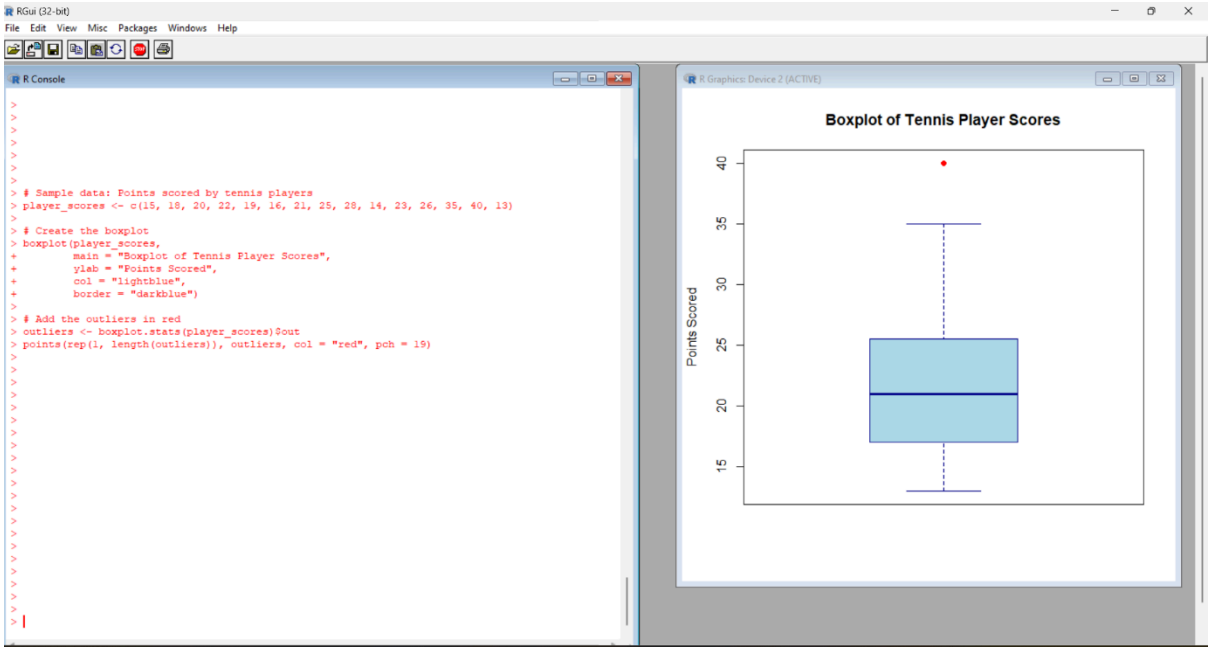
7. Download the Dataset "water" From R dataset Link.Find out whether there is a linear relation between attributes"mortality" and"hardness" by plot function.Fit the Data into the Linear Regression model.Predict the mortality for the hardness=88.

```
> 
> # Create the dataset manually
> water <- data.frame(
+   hardness = c(70, 85, 100, 120, 55, 90, 45, 80, 95, 110),
+   mortality = c(1500, 1600, 1550, 1700, 1400, 1620, 1350, 1580, 1650, 1720)
+ )
> 
> # View the dataset
> head(water)
  hardness mortality
1       70      1500
2       85      1600
3      100      1550
4      120      1700
5       55      1400
6       90      1620
> 
> # Check available datasets in base R
> data()
> 
> # Or use a package like 'datasets' to load some other dataset
> library(datasets)
> data(airquality)  # Example dataset
> head(airquality)
  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
6    28      NA 14.9   66     5   6
> 
```

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



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.



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

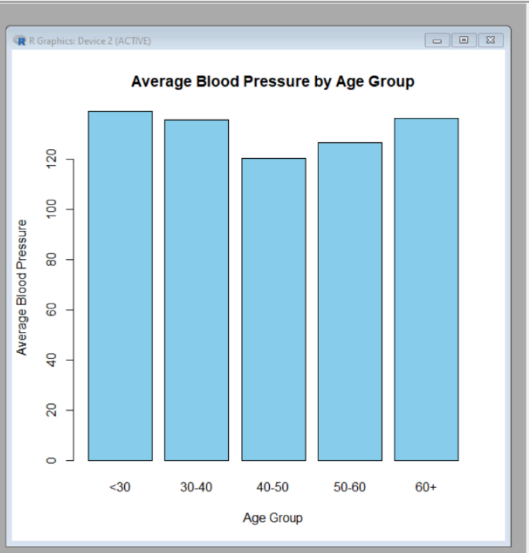
RGui (32-bit)

File Edit View Misc Packages Windows Help

R Console

```
> avg_bp_by_age <- aggregate(BloodPressure ~ AgeGroup, data = diabetes_data, FUN = mean)
>
> # Bar chart with base R
> barplot(avg_bp_by_age$BloodPressure,
+         names.arg = avg_bp_by_age$AgeGroup,
+         main = "Average Blood Pressure by Age Group",
+         xlab = "Age Group",
+         ylab = "Average Blood Pressure",
+         col = "skyblue")
>
>
> # Create a sample dataset
> set.seed(123)  # For reproducibility
> diabetes_data <- data.frame(
+   Age = sample(20:70, 100, replace = TRUE),
+   BloodPressure = sample(80:180, 100, replace = TRUE)
+ )
>
> # Scatter plot with base R
> plot(diabetes_data$Age, diabetes_data$BloodPressure,
+      main = "Scatter Plot of Blood Pressure vs Age",
+      xlab = "Age",
+      ylab = "Blood Pressure",
+      col = "blue",
+      pch = 16)
>
> # Create age groups
> diabetes_data$AgeGroup <- cut(diabetes_data$Age,
+                               breaks = c(0, 30, 40, 50, 60, Inf),
+                               labels = c("<30", "30-40", "40-50", "50-60", "60+"))
>
> # Calculate the average blood pressure in each age group
> avg_bp_by_age <- aggregate(BloodPressure ~ AgeGroup, data = diabetes_data, FUN = mean)
>
> # Bar chart with base R
> barplot(avg_bp_by_age$BloodPressure,
+         names.arg = avg_bp_by_age$AgeGroup,
+         main = "Average Blood Pressure by Age Group",
+         xlab = "Age Group",
+         ylab = "Average Blood Pressure",
+         col = "skyblue")
>
```

R Graphics: Device 2 (ACTIVE)



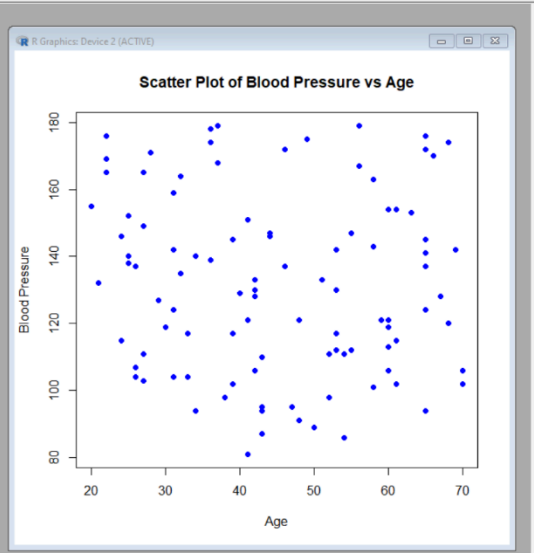
RGui (32-bit)

File Edit View Misc Packages Windows Help

R Console

```
>
>
> # Create a sample dataset
> set.seed(123)  # For reproducibility
> diabetes_data <- data.frame(
+   Age = sample(20:70, 100, replace = TRUE),
+   BloodPressure = sample(80:180, 100, replace = TRUE)
+ )
>
> # View the first few rows of the dataset
> head(diabetes_data)
  Age BloodPressure
1  34           140
2  60           113
3  40           129
4  65           176
5  67           128
6  22           169
>
> # Scatter plot with base R
> plot(diabetes_data$Age, diabetes_data$BloodPressure,
+      main = "Scatter Plot of Blood Pressure vs Age",
+      xlab = "Age",
+      ylab = "Blood Pressure",
+      col = "blue",
+      pch = 16)
>
> # Create age groups
> diabetes_data$AgeGroup <- cut(diabetes_data$Age,
+                               breaks = c(0, 30, 40, 50, 60, Inf),
+                               labels = c("<30", "30-40", "40-50", "50-60", "60+"))
>
> # Calculate the average blood pressure in each age group
> avg_bp_by_age <- aggregate(BloodPressure ~ AgeGroup, data = diabetes_data, FUN = mean)
>
> # Bar chart with base R
> barplot(avg_bp_by_age$BloodPressure,
+         names.arg = avg_bp_by_age$AgeGroup,
+         main = "Average Blood Pressure by Age Group",
+         xlab = "Age Group",
+         ylab = "Average Blood Pressure",
+         col = "skyblue")
```

R Graphics: Device 2 (ACTIVE)



DAY 3

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

@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

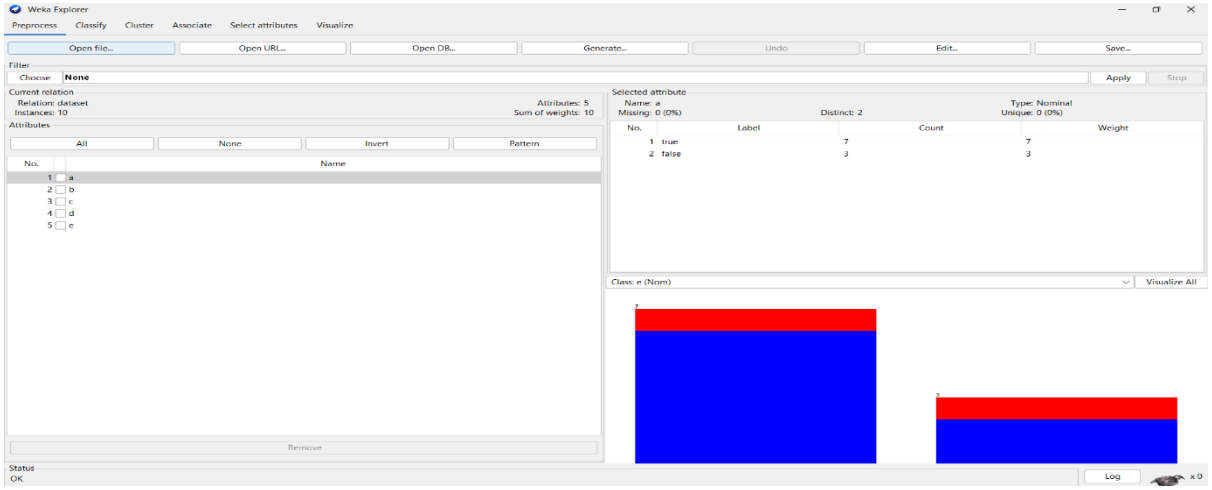
false true false true true

false false true true false

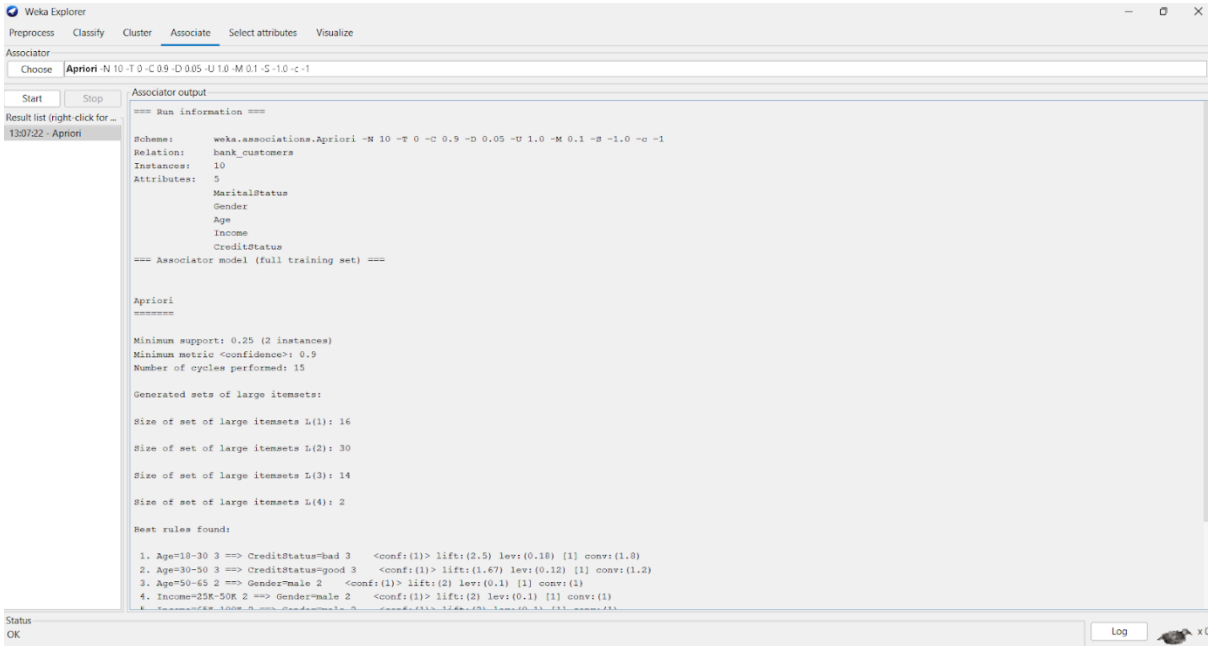
true true true false false

true false false true true

true true false false true



Apriori algorithm:




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


Apriori
=======

Minimum support: 0.35 (3 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 13

Generated sets of large itemsets:

Size of set of large itemsets L(1): 9

Size of set of large itemsets L(2): 19

Size of set of large itemsets L(3): 12

Size of set of large itemsets L(4): 2

Best rules found:

 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 d=false 3 ==> b=true 3     <conf:(1)> lift:(1.67) lev:(0.12) [1] conv:(1.2)
 8. a=true b=false 3 ==> d=true 3     <conf:(1)> lift:(1.67) lev:(0.12) [1] conv:(1.2)
 9. b=false e=true 3 ==> a=true 3     <conf:(1)> lift:(1.43) lev:(0.09) [0] conv:(0.9)
10. a=true b=false 3 ==> e=true 3     <conf:(1)> lift:(1.25) lev:(0.06) [0] conv:(0.6)
```

F growth:

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 0.1
Relation:     dataset
Instances:    10
Attributes:   5
              a
              b
              c
              d
              e
=== Associator model (full training set) ===

FPGrowth found 3 rules (displaying top 3)

1. [b=false, a=false]: 1 ==> [e=false]: 1   <conf:(1)> lift:(5) lev:(0.08) conv:(0.8)
2. [b=false, e=false]: 1 ==> [a=false]: 1   <conf:(1)> lift:(3.33) lev:(0.07) conv:(0.7)
3. [a=false, e=false]: 1 ==> [b=false]: 1   <conf:(1)> lift:(2.5) lev:(0.06) conv:(0.6)
```

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 minsup > 0)?

CODE:

@relation dataset

@attribute Milk{true,false}

@attribute Bread{true,false}

@attribute Diapers{true,false}

@attribute Butter{true,false}

@attribute Beer{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

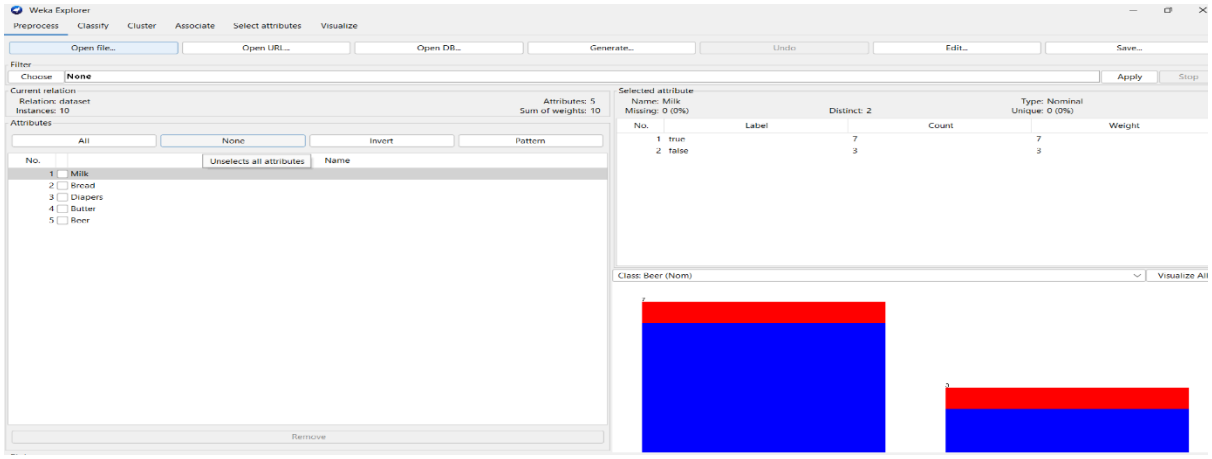
false true false true true

false false true true false

true true true false false

true false false true true

true true false false true



Apriori algorithm:

```
Apriori
=======

Minimum support: 0.35 (3 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 13

Generated sets of large itemsets:

Size of set of large itemsets L(1): 9

Size of set of large itemsets L(2): 19

Size of set of large itemsets L(3): 12

Size of set of large itemsets L(4): 2

Best rules found:

 1. Diapers=false 5 ==> Beer=true 5    <conf:(1)> lift:(1.25) lev:(0.1) [0] conv:(1)
 2. Butter=false 4 ==> Bread=true 4    <conf:(1)> lift:(1.67) lev:(0.16) [1] conv:(1.6)
 3. Bread=false 4 ==> Butter=true 4    <conf:(1)> lift:(1.67) lev:(0.16) [1] conv:(1.6)
 4. Milk=true Diapers=false 4 ==> Beer=true 4    <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.8)
 5. Milk=true Butter=true 4 ==> Beer=true 4    <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.8)
 6. Diapers=false Butter=true 4 ==> Beer=true 4    <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.8)
 7. Milk=true Butter=false 3 ==> Bread=true 3    <conf:(1)> lift:(1.67) lev:(0.12) [1] conv:(1.2)
 8. Milk=true Bread=false 3 ==> Butter=true 3    <conf:(1)> lift:(1.67) lev:(0.12) [1] conv:(1.2)
 9. Bread=false Beer=true 3 ==> Milk=true 3    <conf:(1)> lift:(1.43) lev:(0.09) [0] conv:(0.9)
10. Milk=true Bread=false 3 ==> Beer=true 3    <conf:(1)> lift:(1.25) lev:(0.06) [0] conv:(0.6)
```

F growth:

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 0.1
Relation:     dataset
Instances:    10
Attributes:   5
              Milk
              Bread
              Diapers
              Butter
              Beer
=== Associator model (full training set) ===

FPGrowth found 3 rules (displaying top 3)

1. [Bread=false, Milk=false]: 1 ==> [Beer=false]: 1   <conf:(1)> lift:(5) lev:(0.08) conv:(0.8)
2. [Bread=false, Beer=false]: 1 ==> [Milk=false]: 1   <conf:(1)> lift:(3.33) lev:(0.07) conv:(0.7)
3. [Milk=false, Beer=false]: 1 ==> [Bread=false]: 1   <conf:(1)> lift:(2.5) lev:(0.06) conv:(0.6)
```

3, Bayes classification and descion tree (using training and test data)

Dataset: (sample.arff)

@relation dataset

@attribute age {<=30,31-40,>40}

@attribute income {high,medium,low}

@attribute student {yes,no}

@attribute credit {fair,excellent}

@attribute class {yes,no}

@data

<=30,high,no,fair,no

<=30,high,no,excellent,no

31-40,high,no,fair,yes

>40,medium,no,fair,yes

>40,low,yes,fair,yes

>40,low,yes,excellent,no

31-40,low,yes,excellent,yes

<=30,medium,no,fair,no

<=30,low,yes,fair,yes

>40,medium,yes,fair,yes

<=30,medium,yes,excellent,yes

31-40,medium,no,excellent,yes

31-40,high,yes,fair,yes

>40,medium,no,excellent,no

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

Transactions	Items
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 item sets using Apriori algorithm
- Also draw FP-Growth Tree

@relation dataset

@attribute Monkey{true,false}

@attribute donkey{true,false}

@attribute make{true,false}

@attribute mucky{true,false}

@attribute cookie{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

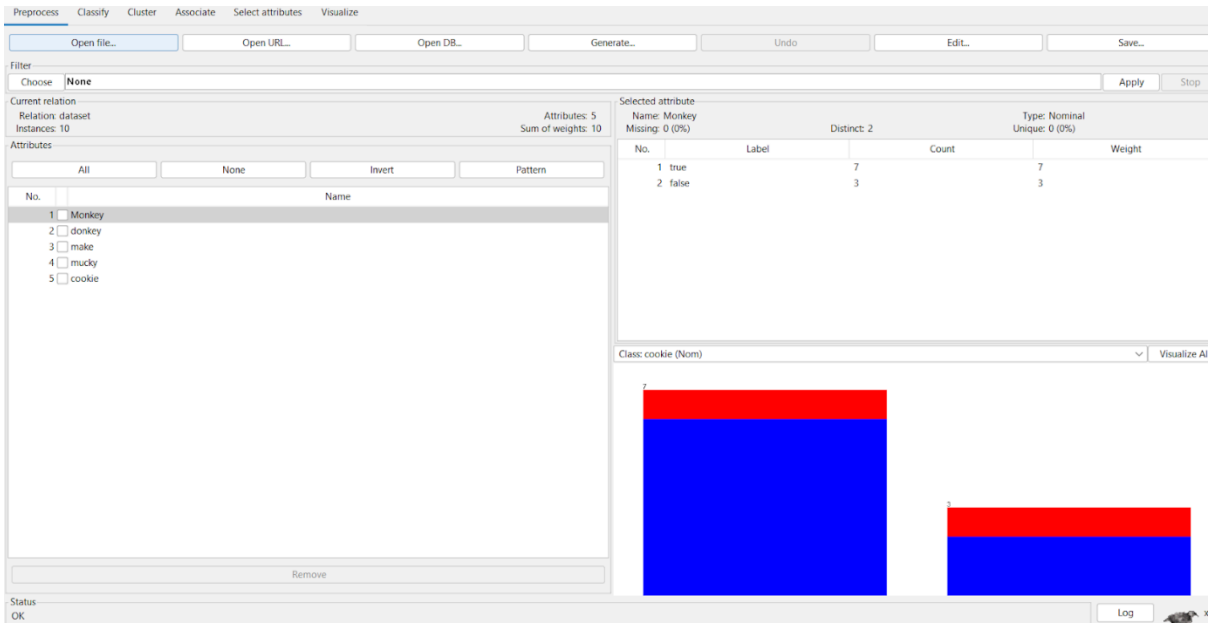
false true false true true

false false true true false

true true true false false

true false false true true

true true false false true



Apriori algorithm:

```
Apriori
=======

Minimum support: 0.35 (3 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 13

Generated sets of large itemsets:

Size of set of large itemsets L(1): 9

Size of set of large itemsets L(2): 19

Size of set of large itemsets L(3): 12

Size of set of large itemsets L(4): 2

Best rules found:

 1. make=false 5 ==> cookie=true 5    <conf:(1)> lift:(1.25) lev:(0.1) [0] conv:(1)
 2. mucky=false 4 ==> donkey=true 4    <conf:(1)> lift:(1.67) lev:(0.16) [1] conv:(1.6)
 3. donkey=false 4 ==> mucky=true 4    <conf:(1)> lift:(1.67) lev:(0.16) [1] conv:(1.6)
 4. Monkey=true make=false 4 ==> cookie=true 4    <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.8)
 5. Monkey=true mucky=true 4 ==> cookie=true 4    <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.8)
 6. make=false mucky=true 4 ==> cookie=true 4    <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.8)
 7. Monkey=true mucky=false 3 ==> donkey=true 3    <conf:(1)> lift:(1.67) lev:(0.12) [1] conv:(1.2)
 8. Monkey=true donkey=false 3 ==> mucky=true 3    <conf:(1)> lift:(1.67) lev:(0.12) [1] conv:(1.2)
 9. donkey=false cookie=true 3 ==> Monkey=true 3    <conf:(1)> lift:(1.43) lev:(0.09) [0] conv:(0.9)
10. Monkey=true donkey=false 3 ==> cookie=true 3    <conf:(1)> lift:(1.25) lev:(0.06) [0] conv:(0.6)
```

F growth:

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 0.1
Relation:     dataset
Instances:    10
Attributes:   5
              Monkey
              donkey
              make
              mucky
              cookie
=== Associator model (full training set) ===

FPGrowth found 3 rules (displaying top 3)

1. [donkey=false, Monkey=false]: 1 ==> [cookie=false]: 1   <conf:(1)> lift:(5) lev:(0.08) conv:(0.8)
2. [donkey=false, cookie=false]: 1 ==> [Monkey=false]: 1   <conf:(1)> lift:(3.33) lev:(0.07) conv:(0.7)
3. [Monkey=false, cookie=false]: 1 ==> [donkey=false]: 1   <conf:(1)> lift:(2.5) lev:(0.06) conv:(0.6)
```

5, Create the dataset using ARFF file format:

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 algorthim ,mention wheather accepted or rejcted.

@relation dataset

@attribute hotdogs{true,false}

@attribute buns{true,false}

@attribute coke{true,false}

@attribute chip{true,false}

@attribute ketchup{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

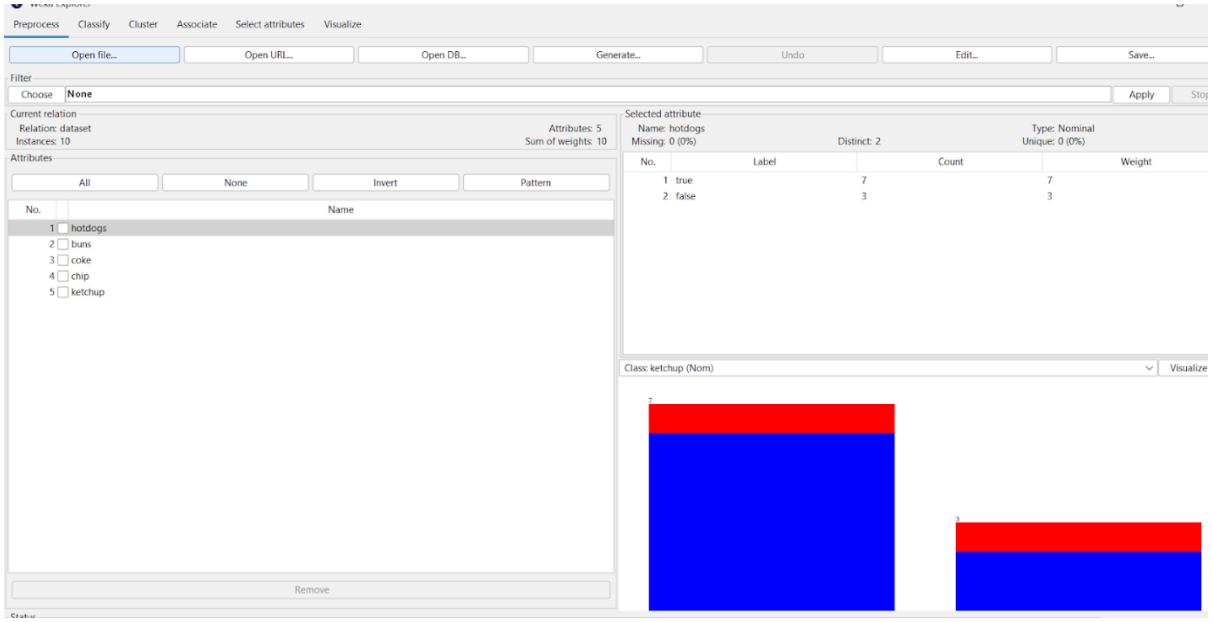
false true false true true

false false true true false

true true true false false

true false false true true

true true false false true



Apriori algorithm:

```
Apriori
=======

Minimum support: 0.35 (3 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 13

Generated sets of large itemsets:

Size of set of large itemsets L(1): 9

Size of set of large itemsets L(2): 19

Size of set of large itemsets L(3): 12

Size of set of large itemsets L(4): 2

Best rules found:

 1. coke=false 5 ==> ketchup=true 5    <conf:(1)> lift:(1.25) lev:(0.1) [0] conv:(1)
 2. chip=false 4 ==> buns=true 4    <conf:(1)> lift:(1.67) lev:(0.16) [1] conv:(1.6)
 3. buns=false 4 ==> chip=true 4    <conf:(1)> lift:(1.67) lev:(0.16) [1] conv:(1.6)
 4. hotdogs=true coke=false 4 ==> ketchup=true 4    <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.8)
 5. hotdogs=true chip=true 4 ==> ketchup=true 4    <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.8)
 6. coke=false chip=true 4 ==> ketchup=true 4    <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.8)
 7. hotdogs=true chip=false 3 ==> buns=true 3    <conf:(1)> lift:(1.67) lev:(0.12) [1] conv:(1.2)
 8. hotdogs=true buns=false 3 ==> chip=true 3    <conf:(1)> lift:(1.67) lev:(0.12) [1] conv:(1.2)
 9. buns=false ketchup=true 3 ==> hotdogs=true 3    <conf:(1)> lift:(1.43) lev:(0.09) [0] conv:(0.9)
10. hotdogs=true buns=false 3 ==> ketchup=true 3    <conf:(1)> lift:(1.25) lev:(0.06) [0] conv:(0.6)
```

F

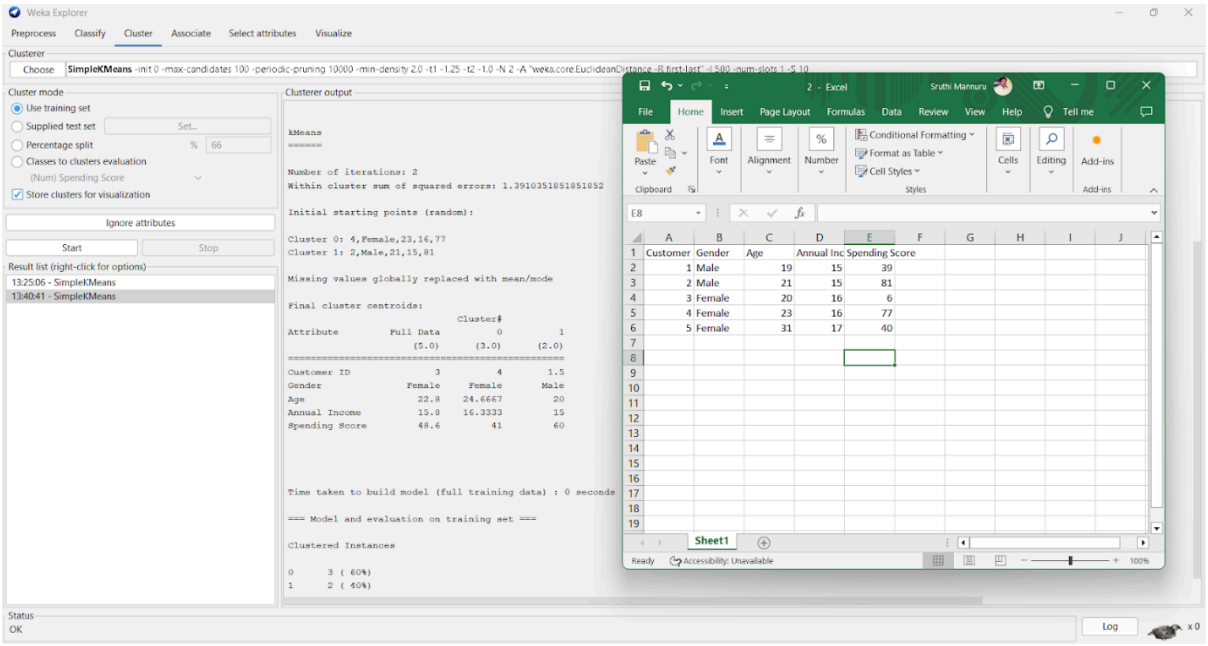
GROWTH:

```
=== Run information ===

Scheme:       weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1
Relation:     dataset
Instances:    10
Attributes:   5
              hotdogs
              buns
              coke
              chip
              ketchup
=== Associator model (full training set) ===

FPGrowth found 3 rules (displaying top 3)

1. [buns=false, hotdogs=false]: 1 ==> [ketchup=false]: 1   <conf:(1)> lift:(5) lev:(0.08) conv:(0.8)
2. [buns=false, ketchup=false]: 1 ==> [hotdogs=false]: 1   <conf:(1)> lift:(3.33) lev:(0.07) conv:(0.7)
3. [hotdogs=false, ketchup=false]: 1 ==> [buns=false]: 1   <conf:(1)> lift:(2.5) lev:(0.06) conv:(0.6)
```

DAY 4

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

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
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

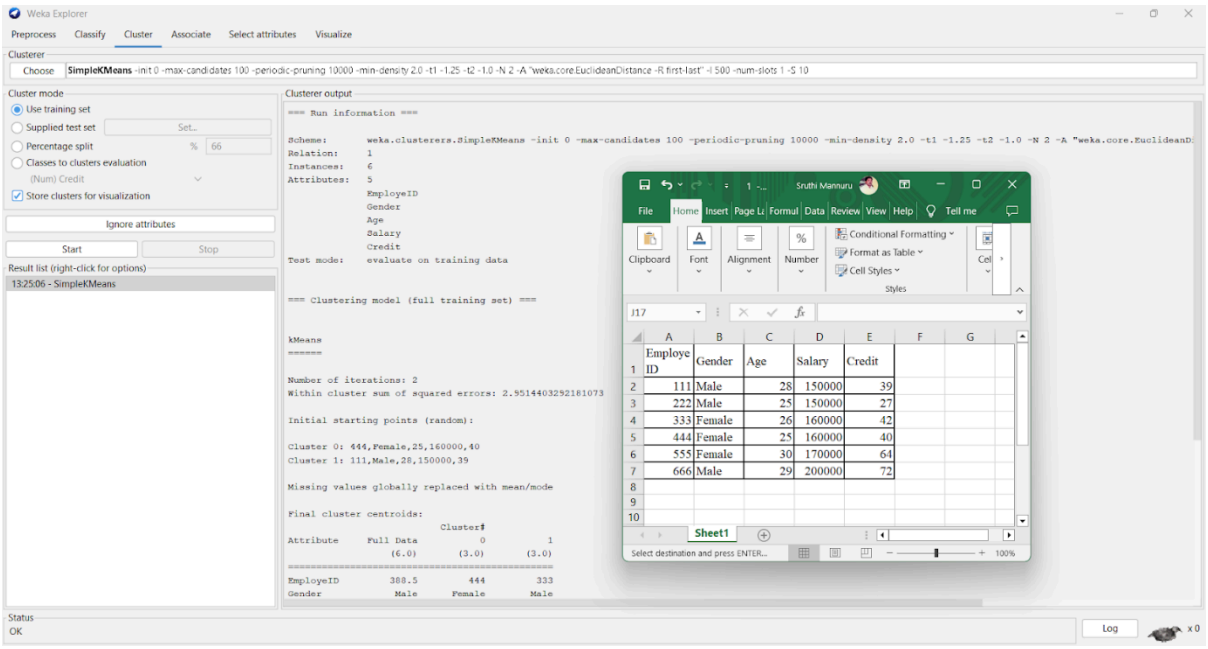


2. Create the following dataset using CSV file format. To perform cluster analysis using K-Means in WEKA. To change the cluster size and plot the graph and illustrate the visualization of cluster.

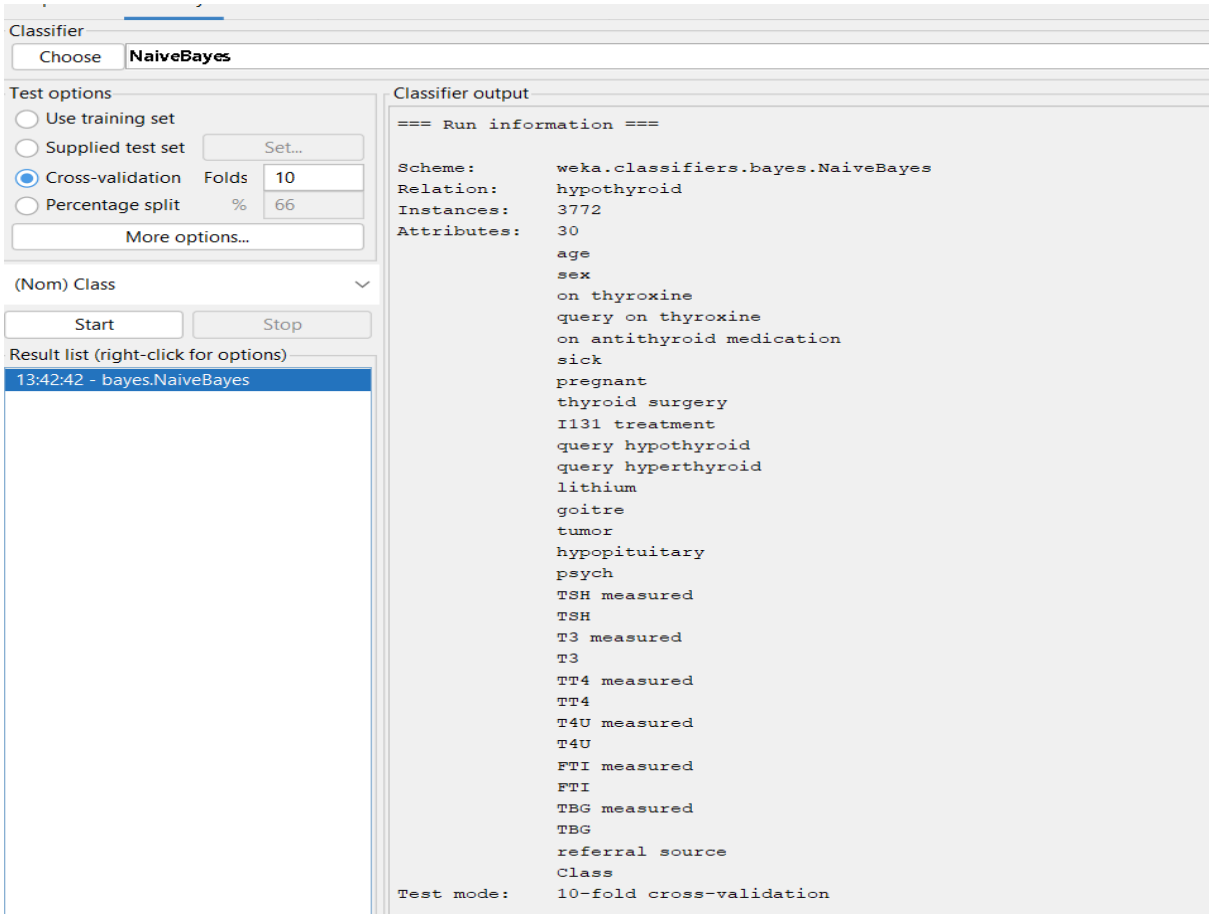
EmployeID	Gender	Age	Salary	Credit
111	Male	28	150000	39
222	Male	25	150000	27
333	Female	26	160000	42
444	Female	25	160000	40
555	Female	30	170000	64

666	Male	29	200000	72

K MEANS CLUSTERING:



3. .Prediction of categorical data using Naïve Bayes classification through WEKA using any datasets. Compare the Naïve Bayes algorithm with SVM using the summary of results given by the classifiers and plot the graph.



Attribute	Class negative (0.92)	compensated_hypothyroid (0.05)	primary_hypothyroid (0.03)	secondary_hypothyroid (0)
age				
mean	51.8081	52.324	50.3348	41.9457
std. dev.	20.3449	19.5296	18.7207	2.4674
weight sum	3480	194	95	2
precision	4.9348	4.9348	4.9348	4.9348
sex				
F	2266.0	146.0	70.0	2.0
M	1078.0	43.0	23.0	2.0
[total]	3344.0	189.0	93.0	4.0
on thyroxine				
f	3027.0	195.0	87.0	3.0
t	456.0	1.0	10.0	1.0
[total]	3483.0	196.0	97.0	4.0
query on thyroxine				
f	3435.0	192.0	96.0	3.0
t	48.0	4.0	1.0	1.0
[total]	3483.0	196.0	97.0	4.0
on antithyroid medication				
f	3440.0	194.0	96.0	3.0
t	43.0	2.0	1.0	1.0
[total]	3483.0	196.0	97.0	4.0
sick				
f	3346.0	184.0	96.0	3.0
t	137.0	12.0	1.0	1.0
[total]	3483.0	196.0	97.0	4.0

```
Time taken to build model: 0.04 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3594               95.281  %
Incorrectly Classified Instances       178                4.719  %
Kappa statistic                          0.6008
Mean absolute error                      0.0357
Root mean squared error                  0.1382
Relative absolute error                 48.9161 %
Root relative squared error             72.5471 %
Total Number of Instances             3772

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.993    0.485    0.961      0.993    0.977      0.645    0.930     0.992     negative
                 0.320    0.007    0.721      0.320    0.443      0.463    0.900     0.534     compensated_hypothyroid
                 0.800    0.003    0.874      0.800    0.835      0.832    0.995     0.865     primary_hypothyroid
                 0.000    0.001    0.000      0.000    0.000      -0.001   0.288     0.001     secondary_hypothyroid
Weighted Avg.    0.953    0.448    0.946      0.953    0.945      0.640    0.929     0.965

=== Confusion Matrix ===

    a    b    c    d   <-- classified as
 3456   14    9    2 |    a = negative
  130   62    2    0 |    b = compensated_hypothyroid
    9   10   76    0 |    c = primary_hypothyroid
    2    0    0    0 |    d = secondary hypothyroid
```

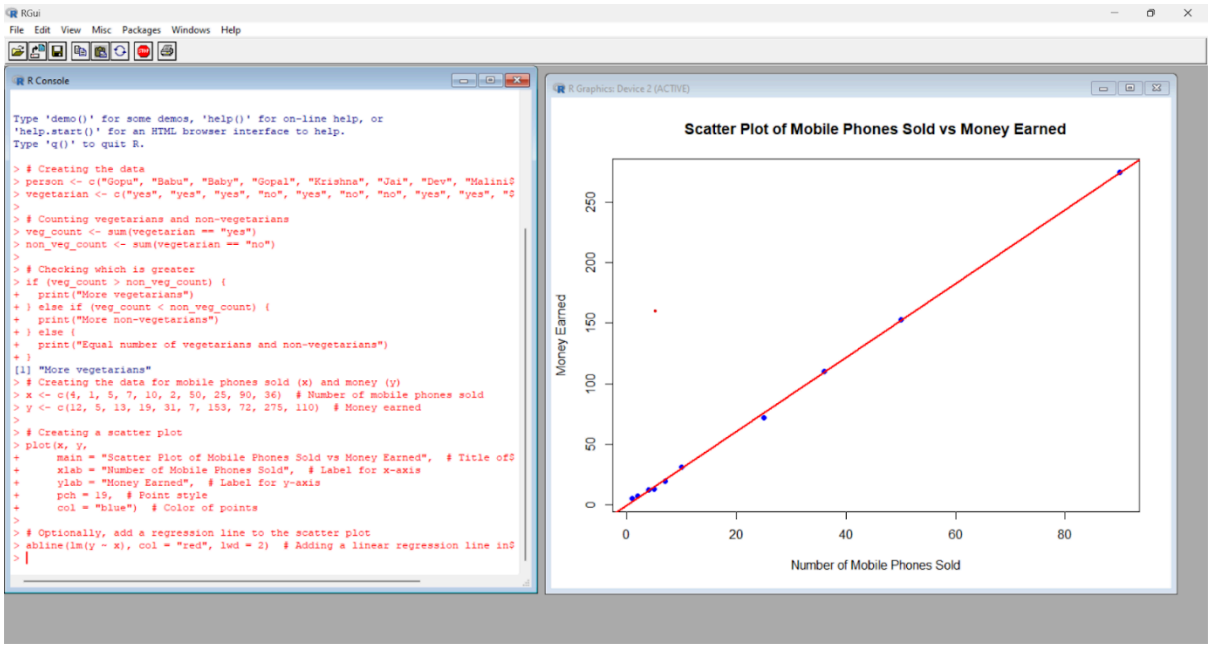
4.The following list of persons with vegetarian or not details given in the table. How will you find out how many of them are vegetarian and how many of them are non-vegetarian? Which type of the person total count is greater value?

Person	Gopu	Babu	Baby	Gopal	Krishna	Jai	Dev	Malini	Hema	Anu
Vegetarian	yes	yes	yes	no	yes	no	no	yes	yes	yes

```
> # Defining the vectors for persons and their vegetarian status
> persons <- c("Gopu", "Babu", "Baby", "Gopal", "Krishna", "Jai", "Dev", "Malini", "Hema", "Anu")
> vegetarian_status <- c("yes", "yes", "yes", "no", "yes", "no", "no", "yes", "yes", "yes")
>
> # Counting how many are vegetarian
> vegetarian_count <- sum(vegetarian_status == "yes")
>
> # Counting how many are non-vegetarian
> non_vegetarian_count <- sum(vegetarian_status == "no")
>
> # Output the counts
> cat("Number of vegetarians:", vegetarian_count, "\n")
Number of vegetarians: 7
> cat("Number of non-vegetarians:", non_vegetarian_count, "\n")
Number of non-vegetarians: 3
>
> # Find which count is greater
> if (vegetarian_count > non_vegetarian_count) {
+   cat("Vegetarians are greater in number.")
+ } else if (vegetarian_count < non_vegetarian_count) {
+   cat("Non-vegetarians are greater in number.")
+ } else {
+   cat("The number of vegetarians and non-vegetarians is equal.")
+ }
Vegetarians are greater in number.> 4
[1] 4
```

5.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



6.Generate rules using FP growth algorithm using the given dataset which has the following transactions with items purchased: Consider the values as support=50% and confidence=75%.

Transaction ID	Items Purchased
1	Bread, Cheese, Egg, Juice
2	Bread, Cheese, Juice
3	Bread, Milk, Yogurt
4	Bread, Juice, Milk
5	Cheese, Juice, Milk

@relation dataset

@attribute bread{true,false}

@attribute cheese{true,false}

@attribute egg{true,false}

@attribute jucie{true,false}

@attribute yogurat{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

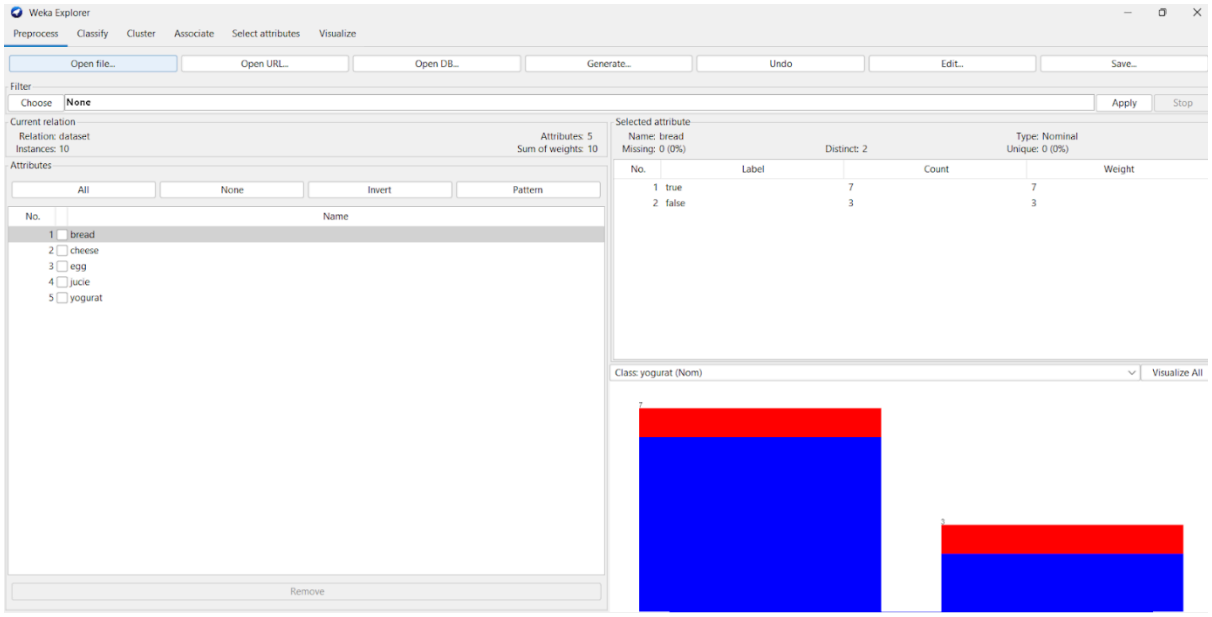
false true false true true

false false true true false

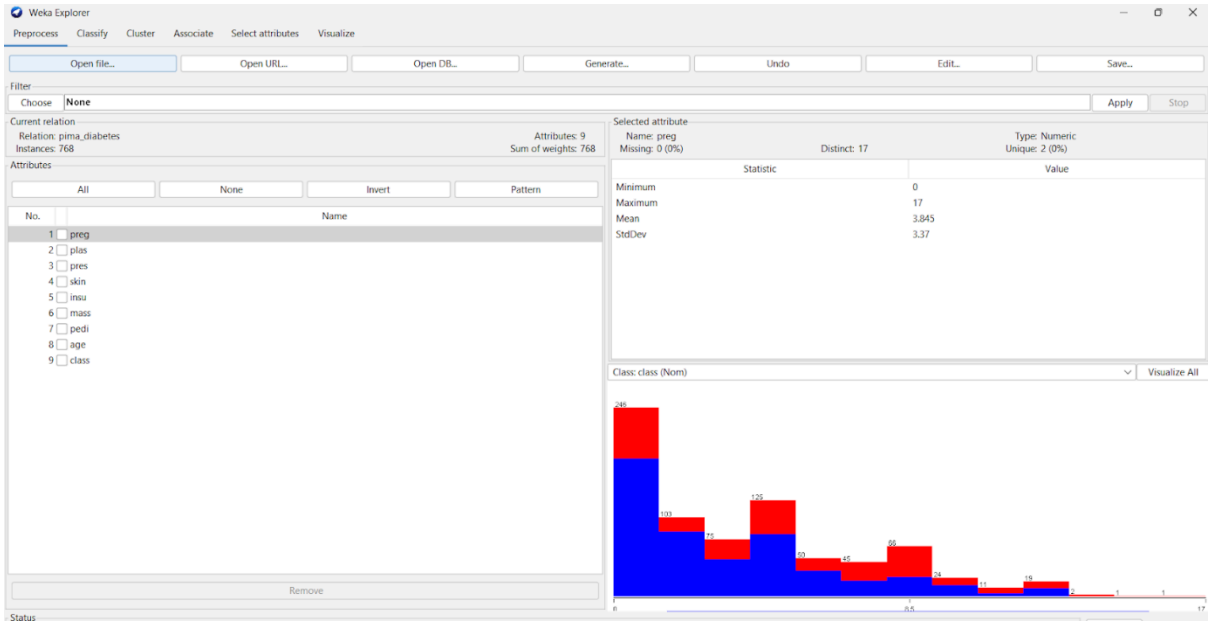
true true true false false

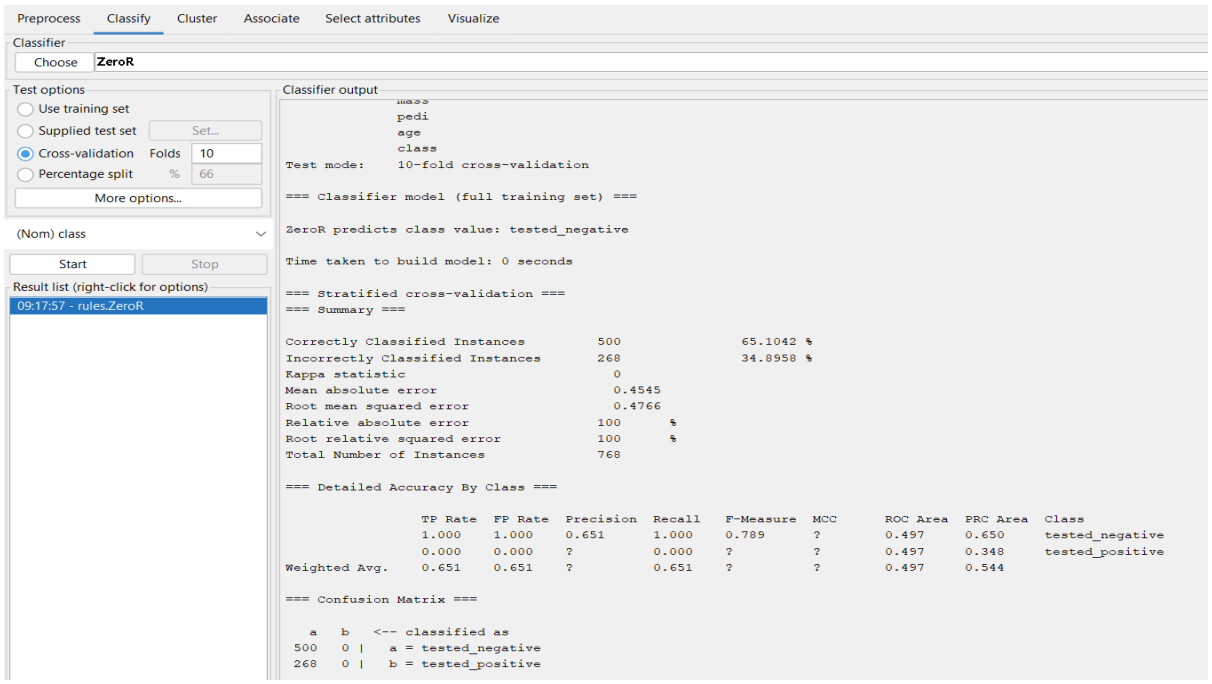
true false false true true

true true false false true



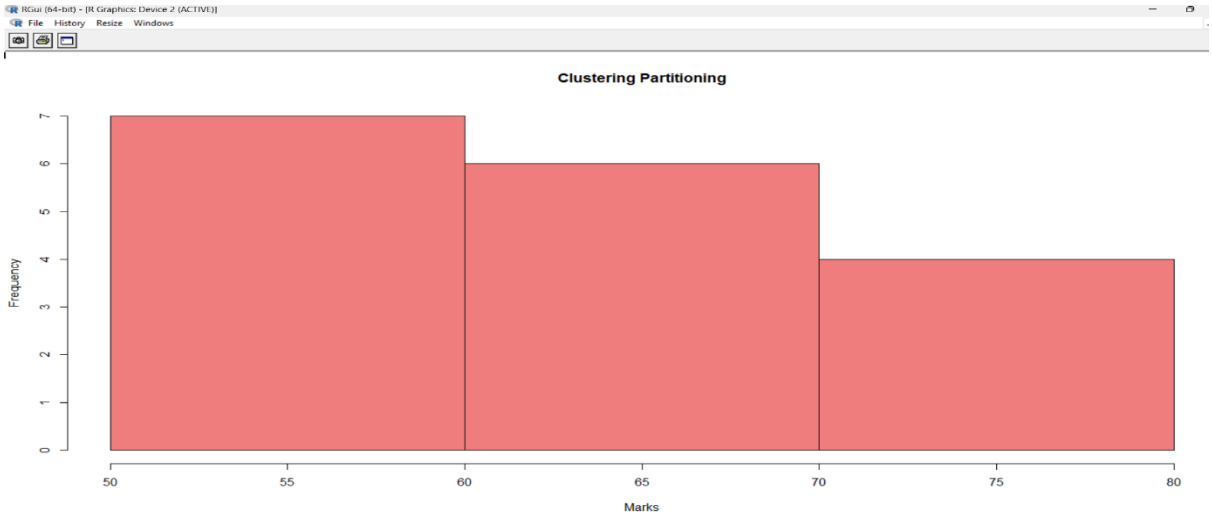
7.


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Current relation
Relation: pima_diabetes
Instances: 768
Attributes: 9
Sum of weights: 768
Attributes
All
None
Invert
Pattern
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
Status


Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
ZeroR
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)
09:17:57 - rules.ZeroR
Classifier output
mass
pedi
age
class
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
ZeroR predicts class value: tested_negative
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 500 65.1042 %
Incorrectly Classified Instances 268 34.8958 %
Kappa statistic 0
Mean absolute error 0.4545
Root mean squared error 0.4766
Relative absolute error 100 %
Root relative squared error 100 %
Total Number of Instances 768
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
1.000 1.000 0.651 1.000 0.789 ? 0.497 0.650 tested_negative
0.000 0.000 ? 0.000 ? ? 0.497 0.348 tested_positive
Weighted Avg. 0.651 0.651 ? 0.651 ? ? 0.497 0.544
=== Confusion Matrix ===
a b <-- classified as
500 0 | a = tested_negative
268 0 | b = tested_positive

8.

```
> # Define the marks scored by the student
> marks <- c(55, 60, 71, 63, 55, 65, 50, 55, 58, 59, 61, 63, 65, 67, 71, 72, 75)
>
> # (a) Equal-frequency (equi-depth) partitioning
> # Divide the data into three equal-sized bins
> equal_freq_bins <- cut(marks, breaks=quantile(marks, probs=seq(0, 1, by=1/3)), include.lowest=TRUE)
>
> # Plot histogram for equal-frequency partitioning
> hist(marks, breaks=quantile(marks, probs=seq(0, 1, by=1/3)), main="Equal-frequency (Equi-depth) Partitioning",
+      xlab="Marks", col="lightblue", border="black")
>
> # (b) Equal-width partitioning
> # Determine the bin width based on the range of marks divided into 3 bins
> range_marks <- range(marks)
> width <- (range_marks[2] - range_marks[1]) / 3
> equal_width_bins <- cut(marks, breaks=seq(range_marks[1], range_marks[2], by=width), include.lowest=TRUE)
>
> # Plot histogram for equal-width partitioning
> hist(marks, breaks=seq(range_marks[1], range_marks[2], by=width), main="Equal-width Partitioning",
+      xlab="Marks", col="lightgreen", border="black")
>
> # (c) Clustering (using k-means clustering)
> set.seed(123)  # For reproducibility
> clusters <- kmeans(marks, centers=3)
>
> # Plot histogram for clustering
> hist(marks, breaks=3, main="Clustering Partitioning",
```



9

10.

@relation dataset

@attribute sony{true,false}

@attribute bpl{true,false}

@attribute lg{true,false}

@attribute samsung{true,false}

@attribute onida{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

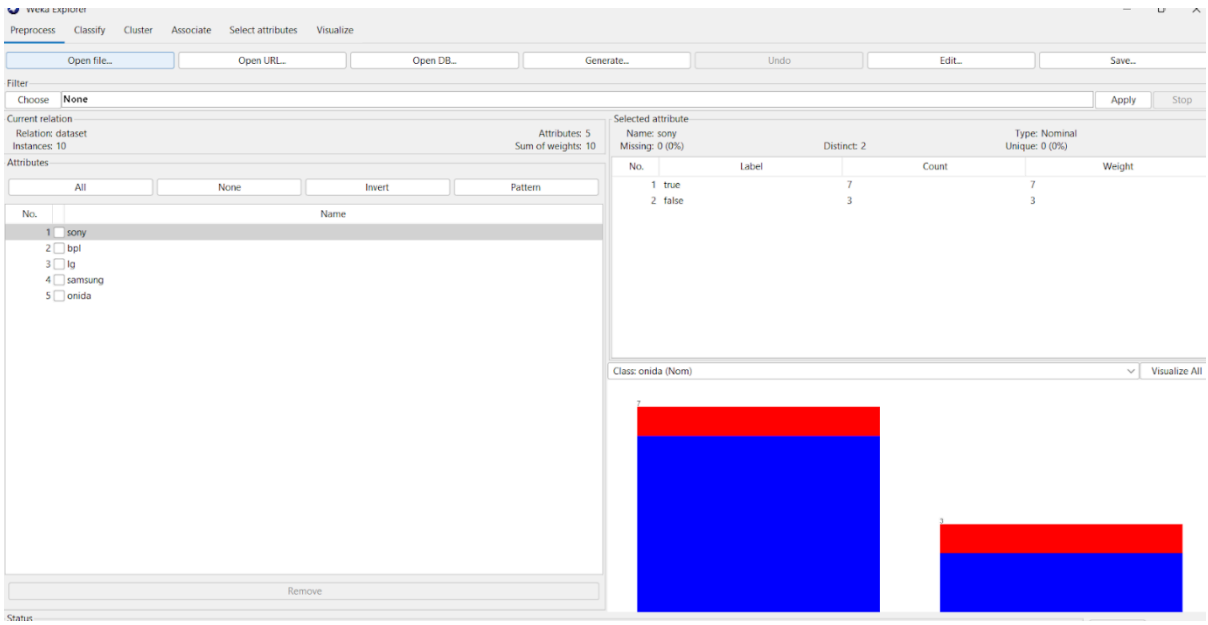
false true false true true

false false true true false

true true true false false

true false false true true

true true false false true



```
Apriori
=======

Minimum support: 0.35 (3 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 13

Generated sets of large itemsets:

Size of set of large itemsets L(1): 9

Size of set of large itemsets L(2): 19

Size of set of large itemsets L(3): 12

Size of set of large itemsets L(4): 2

Best rules found:

 1. lg=false 5 ==> onida=true 5    <conf:(1)> lift:(1.25) lev:(0.1) [0] conv:(1)
 2. samsung=false 4 ==> bpl=true 4    <conf:(1)> lift:(1.67) lev:(0.16) [1] conv:(1.6)
 3. bpl=false 4 ==> samsung=true 4    <conf:(1)> lift:(1.67) lev:(0.16) [1] conv:(1.6)
 4. sony=true lg=false 4 ==> onida=true 4    <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.8)
 5. sony=true samsung=true 4 ==> onida=true 4    <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.8)
 6. lg=false samsung=true 4 ==> onida=true 4    <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.8)
 7. sony=true samsung=false 3 ==> bpl=true 3    <conf:(1)> lift:(1.67) lev:(0.12) [1] conv:(1.2)
 8. sony=true bpl=false 3 ==> samsung=true 3    <conf:(1)> lift:(1.67) lev:(0.12) [1] conv:(1.2)
 9. bpl=false onida=true 3 ==> sony=true 3    <conf:(1)> lift:(1.43) lev:(0.09) [0] conv:(0.9)
10. sony=true bpl=false 3 ==> onida=true 3    <conf:(1)> lift:(1.25) lev:(0.06) [0] conv:(0.6)
```

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 0.1
Relation:     dataset
Instances:    10
Attributes:   5
              sony
              bpl
              lg
              samsung
              onida
=== Associator model (full training set) ===

FPGrowth found 3 rules (displaying top 3)

1. [bpl=false, sony=false]: 1 ==> [onida=false]: 1   <conf:(1)> lift:(5) lev:(0.08) conv:(0.8)
2. [bpl=false, onida=false]: 1 ==> [sony=false]: 1   <conf:(1)> lift:(3.33) lev:(0.07) conv:(0.7)
3. [sony=false, onida=false]: 1 ==> [bpl=false]: 1   <conf:(1)> lift:(2.5) lev:(0.06) conv:(0.6)
```

11.

```
<environment: namespace:base>
> # Given data
> strike_rates <- c(100, 70, 60, 90, 90)
>
> # Min-max normalization
> min_val <- min(strike_rates)
> max_val <- max(strike_rates)
> min_max_norm <- (strike_rates - min_val) / (max_val - min_val)
> min_max_norm
[1] 1.00 0.25 0.00 0.75 0.75
> q
function (save = "default", status = 0, runLast = TRUE)
.Internal(quit(save, status, runLast))
<bytecode: 0x00000184379544e0>
<environment: namespace:base>
> # Z-score normalization
> mean_val <- mean(strike_rates)
> sd_val <- sd(strike_rates)
> z_score_norm <- (strike_rates - mean_val) / sd_val
> z_score_norm
[1]  1.0954451 -0.7302967 -1.3388774  0.4868645  0.4868645
> q
function (save = "default", status = 0, runLast = TRUE)
.Internal(quit(save, status, runLast))
<bytecode: 0x00000184379544e0>
<environment: namespace:base>
> # Z-score normalization using MAD
> mad_val <- mean(abs(strike_rates - mean_val))
> z_mad_norm <- (strike_rates - mean_val) / mad_val
> z_mad_norm
[1]  1.3235294 -0.8823529 -1.6176471  0.5882353  0.5882353
```

12.

```
> q
function (save = "default", status = 0, runLast = TRUE)
.Internal(quit(save, status, runLast))
<bytecode: 0x00001843795544e0>
<environment: namespace:base>
> # Given data
> avg_speed <- c(78, 81, 82, 74, 83, 82, 77, 80, 70)
> total_time <- c(39, 37, 36, 42, 35, 36, 40, 38, 46)
>
> # a) Calculate Standard Deviation
> sd_avg_speed <- sd(avg_speed)
> sd_total_time <- sd(total_time)
>
> # b) Calculate Variance
> var_avg_speed <- var(avg_speed)
> var_total_time <- var(total_time)
>
> # Output the results
> cat("Standard Deviation of AvgSpeed:", sd_avg_speed, "\n")
Standard Deviation of AvgSpeed: 4.304391
> cat("Standard Deviation of TotalTime:", sd_total_time, "\n")
Standard Deviation of TotalTime: 3.492054
> cat("Variance of AvgSpeed:", var_avg_speed, "\n")
Variance of AvgSpeed: 18.52778
> cat("Variance of TotalTime:", var_total_time, "\n")
Variance of TotalTime: 12.19444
>
```

13.

@relation dataset

@attribute Monkey{true,false}

@attribute donkey{true,false}

@attribute make{true,false}

@attribute mucky{true,false}

@attribute cookie{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

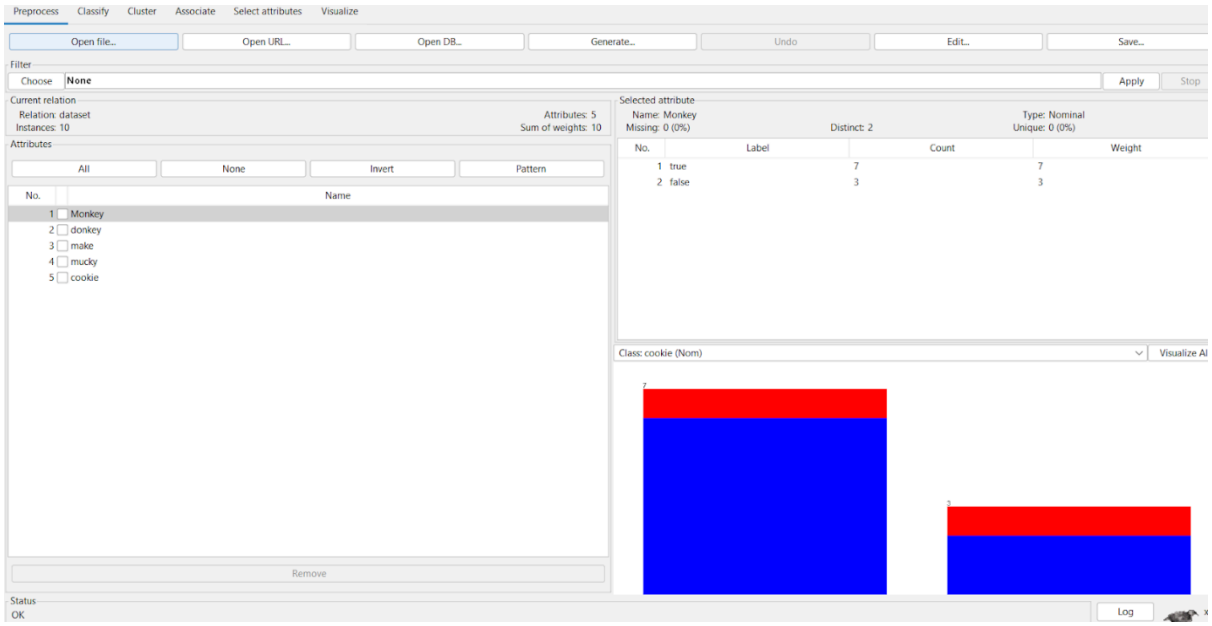
false true false true true

false false true true false

true true true false false

true false false true true

true true false false true




```
Apriori
=======

Minimum support: 0.35 (3 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 13

Generated sets of large itemsets:

Size of set of large itemsets L(1): 9

Size of set of large itemsets L(2): 19

Size of set of large itemsets L(3): 12

Size of set of large itemsets L(4): 2

Best rules found:

 1. Diapers=false 5 ==> Beer=true 5     <conf:(1)> lift:(1.25) lev:(0.1) [0] conv:(1)
 2. Butter=false 4 ==> Bread=true 4     <conf:(1)> lift:(1.67) lev:(0.16) [1] conv:(1.6)
 3. Bread=false 4 ==> Butter=true 4     <conf:(1)> lift:(1.67) lev:(0.16) [1] conv:(1.6)
 4. Milk=true Diapers=false 4 ==> Beer=true 4     <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.8)
 5. Milk=true Butter=true 4 ==> Beer=true 4     <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.8)
 6. Diapers=false Butter=true 4 ==> Beer=true 4     <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.8)
 7. Milk=true Butter=false 3 ==> Bread=true 3     <conf:(1)> lift:(1.67) lev:(0.12) [1] conv:(1.2)
 8. Milk=true Bread=false 3 ==> Butter=true 3     <conf:(1)> lift:(1.67) lev:(0.12) [1] conv:(1.2)
 9. Bread=false Beer=true 3 ==> Milk=true 3     <conf:(1)> lift:(1.43) lev:(0.09) [0] conv:(0.9)
10. Milk=true Bread=false 3 ==> Beer=true 3     <conf:(1)> lift:(1.25) lev:(0.06) [0] conv:(0.6)
```

Associator output

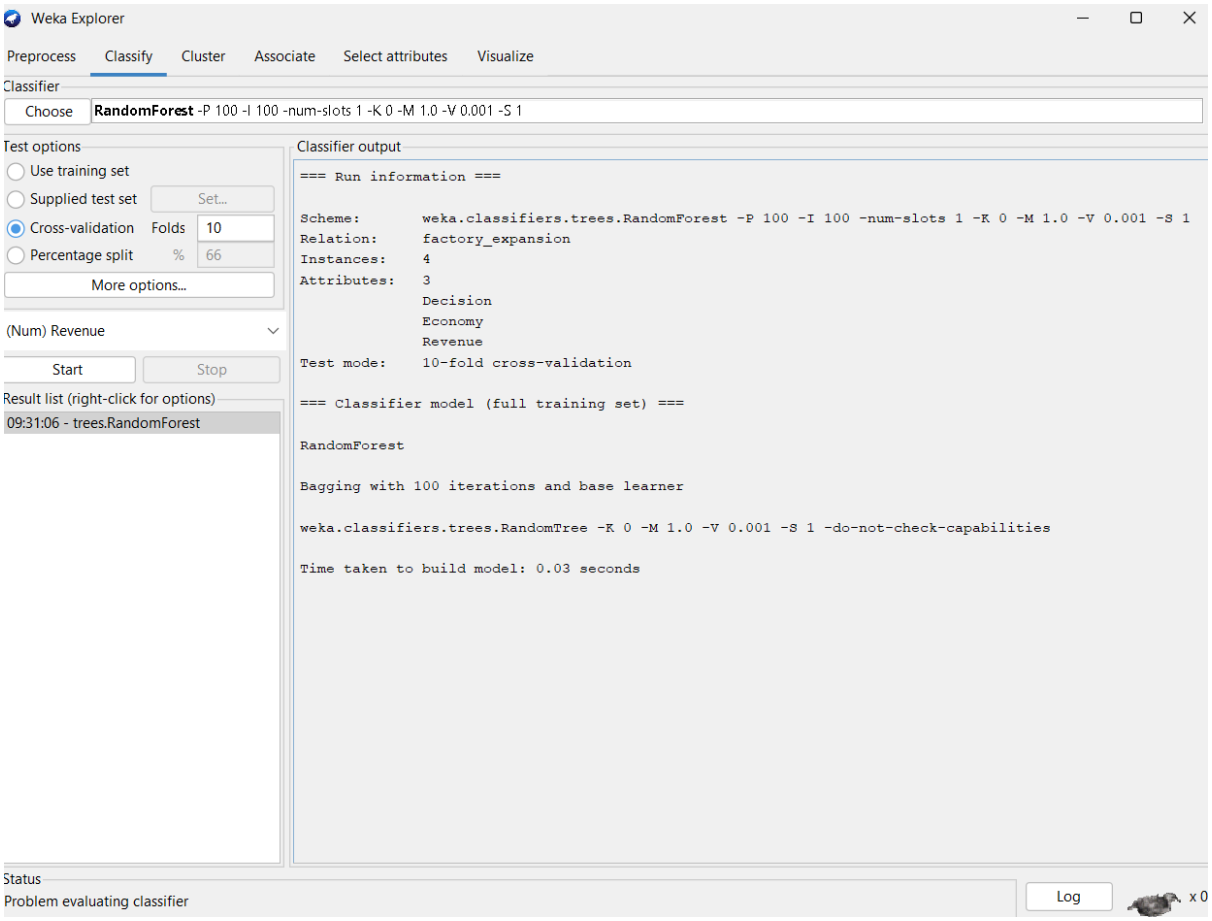
```
=== Run information ===

Scheme:       weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1
Relation:     dataset
Instances:    10
Attributes:   5
              Monkey
              donkey
              make
              mucky
              cookie
=== Associator model (full training set) ===

FPGrowth found 3 rules (displaying top 3)

1. [donkey=false, Monkey=false]: 1 ==> [cookie=false]: 1   <conf:(1)> lift:(5) lev:(0.08) conv:(0.8)
2. [donkey=false, cookie=false]: 1 ==> [Monkey=false]: 1   <conf:(1)> lift:(3.33) lev:(0.07) conv:(0.7)
3. [Monkey=false, cookie=false]: 1 ==> [donkey=false]: 1   <conf:(1)> lift:(2.5) lev:(0.06) conv:(0.6)
```

LAB QUESTIONS BANK

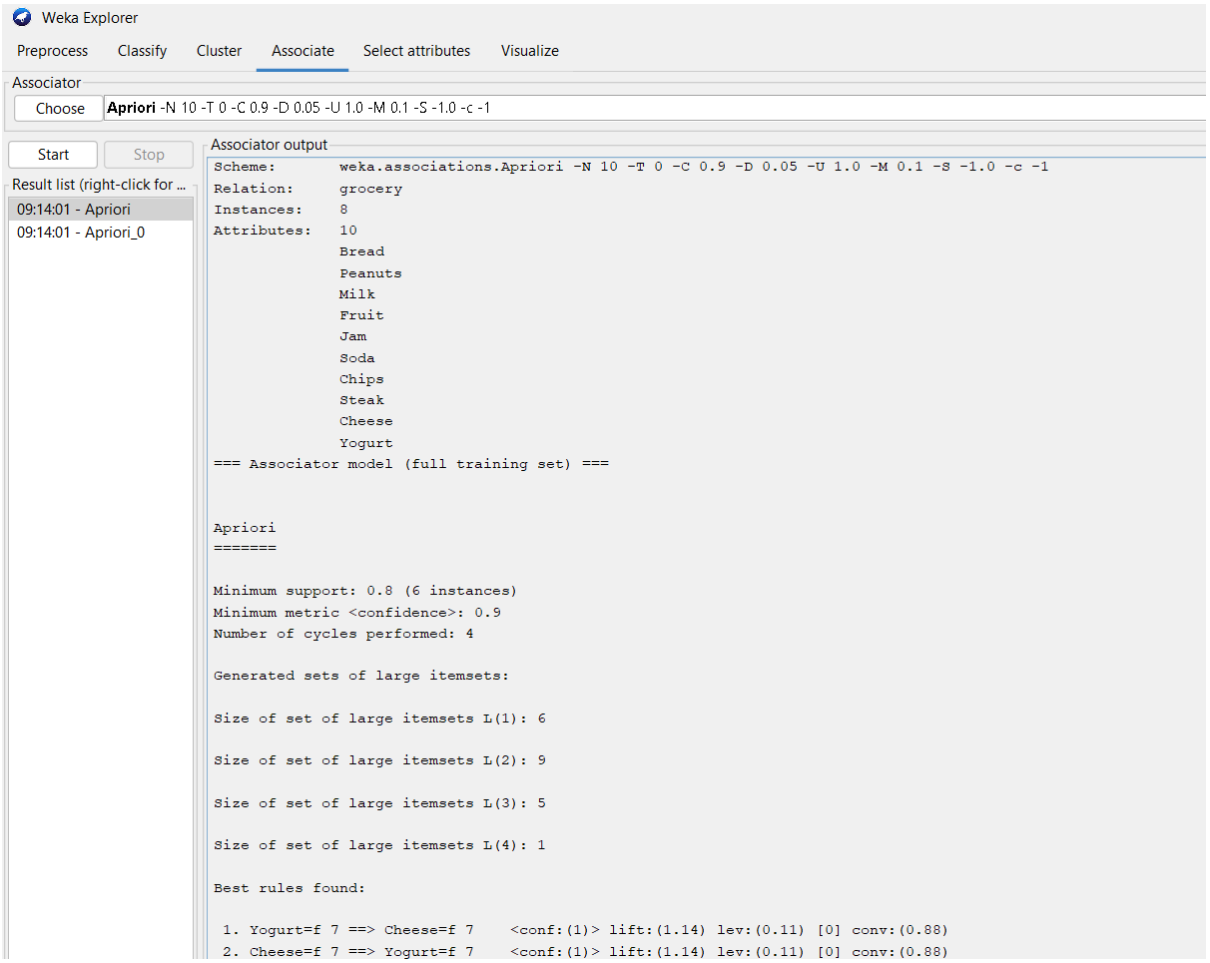
1.

```
> # Given data
> sales_prices <- c(5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215)
>
> # (a) Equal-frequency (equi-depth) partitioning
> # Split data into three bins with approximately equal numbers of records.
> equal_frequency_bins <- split(sales_prices, cut(seq_along(sales_prices), breaks=3, labels=FALSE))
> equal_frequency_bins
$`1`
[1]  5 10 11 13

$`2`
[1] 15 35 50 55

$`3`
[1]  72  92 204 215

>
> # (b) Equal-width partitioning
> # Define the width of each bin
> min_value <- min(sales_prices)
> max_value <- max(sales_prices)
> num_bins <- 3
> bin_width <- (max_value - min_value) / num_bins
>
> # Define the bin edges
> bin_edges <- seq(min_value, max_value, by = bin_width)
>
> # Split the data based on equal-width bins
> equal_width_bins <- split(sales_prices, cut(sales_prices, breaks = bin_edges, include.lowest = TRUE))
> equal_width_bins
$`[5,75]`
[1]  5 10 11 13 15 35 50 55 72

$`(75,145]`
[1] 92

$`(145,215]`
[1] 204 215
```

2.


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Num) Revenue
Start
Stop
Result list (right-click for options)
09:31:06 - trees.RandomForest
Classifier output
=== Run information ===
Scheme: weka.classifiers.trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1
Relation: factory_expansion
Instances: 4
Attributes: 3
Decision
Economy
Revenue
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
RandomForest
Bagging with 100 iterations and base learner
weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities
Time taken to build model: 0.03 seconds
Status
Problem evaluating classifier
Log
x 0

3.

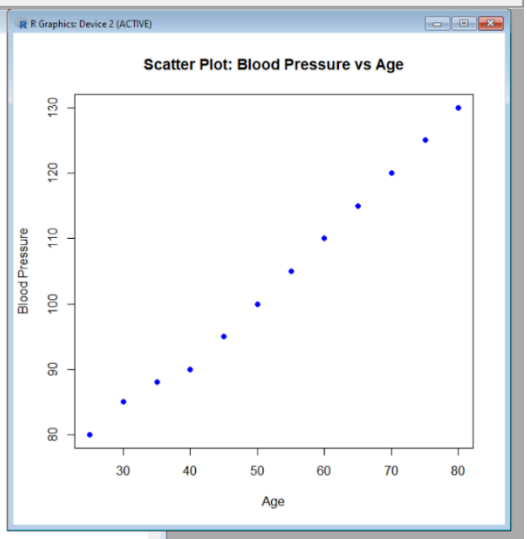

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Start
Stop
Result list (right-click for ...
09:14:01 - Apriori
09:14:01 - Apriori_0
Associator output
Scheme: weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation: grocery
Instances: 8
Attributes: 10
Bread
Peanuts
Milk
Fruit
Jam
Soda
Chips
Steak
Cheese
Yogurt
=== Associator model (full training set) ===
Apriori
=======
Minimum support: 0.8 (6 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 4
Generated sets of large itemsets:
Size of set of large itemsets L(1): 6
Size of set of large itemsets L(2): 9
Size of set of large itemsets L(3): 5
Size of set of large itemsets L(4): 1
Best rules found:
1. Yogurt=f 7 ==> Cheese=f 7 <conf:(1)> lift:(1.14) lev:(0.11) [0] conv:(0.88)
2. Cheese=f 7 ==> Yogurt=f 7 <conf:(1)> lift:(1.14) lev:(0.11) [0] conv:(0.88)

4.

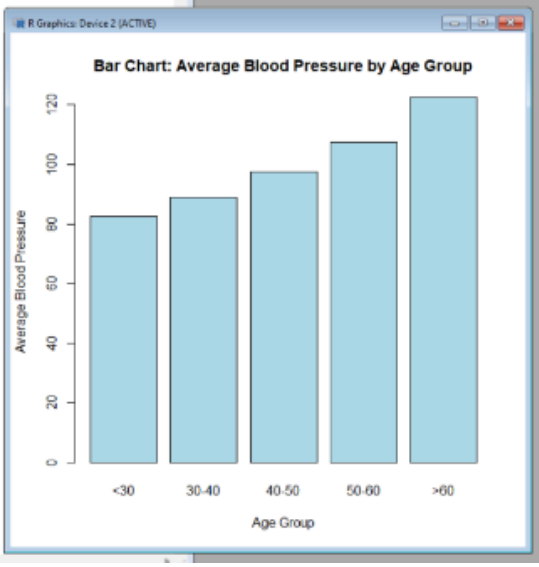
```
> data <- c(200, 300, 400, 600, 1000)
>
> # (a) Min-Max Normalization (min=0, max=1)
> min_max_normalization <- function(x) {
+   (x - min(x)) / (max(x) - min(x))
+ }
>
> min_max <- min_max_normalization(data)
> cat("Min-Max Normalization:\n", min_max, "\n\n")
Min-Max Normalization:
 0 0.125 0.25 0.5 1

>
> # (b) Z-Score Normalization
> z_score_normalization <- function(x) {
+   (x - mean(x)) / sd(x)
+ }
>
> z_score <- z_score_normalization(data)
> cat("Z-Score Normalization:\n", z_score, "\n\n")
Z-Score Normalization:
 -0.9486833 -0.6324555 -0.3162278 0.3162278 1.581139

>
> # (c) Z-Score Normalization using Mean Absolute Deviation (MAD)
> mad_normalization <- function(x) {
+   mad_value <- mean(abs(x - mean(x)))
+   (x - mean(x)) / mad_value
+ }
>
> mad_z_score <- mad_normalization(data)
> cat("Z-Score Normalization with MAD:\n", mad_z_score, "\n\n")
Z-Score Normalization with MAD:
 -1.25 -0.8333333 -0.4166667 0.4166667 2.083333

>
> # (d) Normalization by Decimal Scaling
> decimal_scaling_normalization <- function(x) {
+   j <- max(nchar(abs(as.integer(x))))  # Number of digits in the max absolute value
+   x / (10^j)
+ }
```

5.

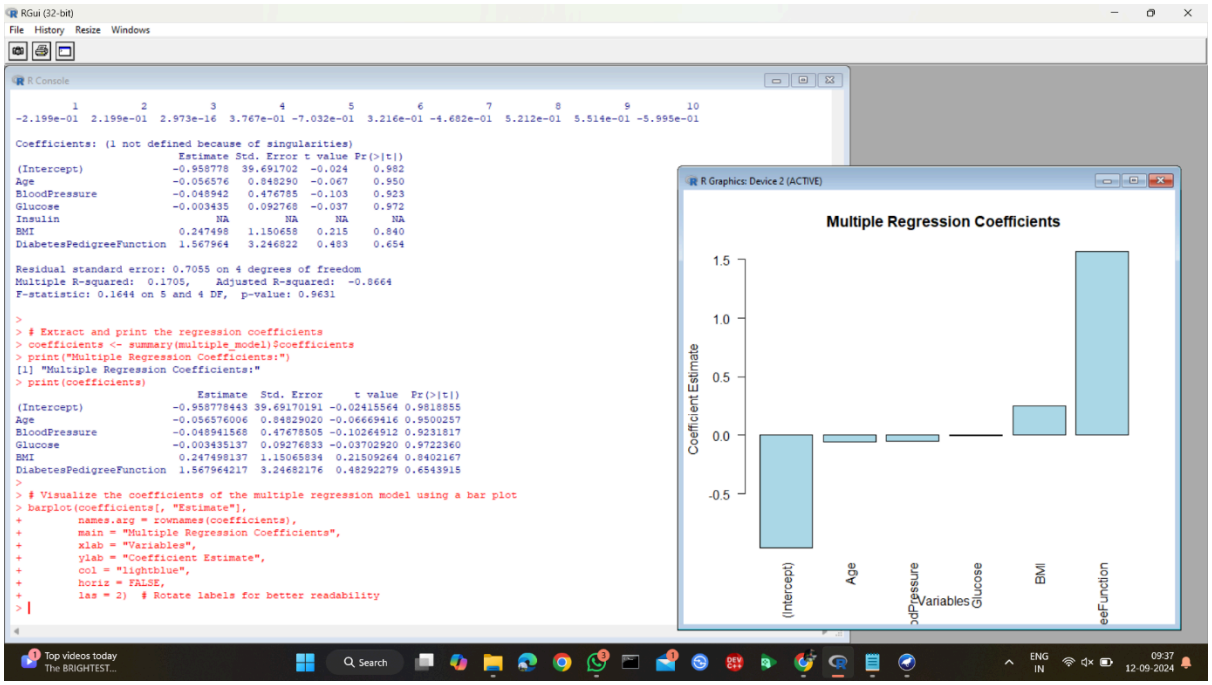
```
> # Creating a simulated diabetes dataset
> diabetes_data <- data.frame(
+   Age = c(25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80),
+   BloodPressure = c(80, 85, 88, 90, 95, 100, 105, 110, 115, 120, 125, 130),
+   Glucose = c(110, 120, 140, 160, 170, 180, 190, 200, 210, 220, 230, 240),
+   Insulin = c(85, 90, 100, 120, 130, 140, 150, 160, 170, 180, 190, 200),
+   BMI = c(22.5, 23.0, 24.5, 26.0, 28.5, 30.0, 32.5, 34.0, 36.0, 38.0, 40.0, 42.0),
+   DiabetesPedigreeFunction = c(0.2, 0.3, 0.4, 0.5, 0.45, 0.35, 0.55, 0.65, 0.75, 0.85, 0.9, 1.0),
+   Outcome = c(0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1)
+ )
>
> # Step 1: Saving the simulated dataset as a CSV file
> write.csv(diabetes_data, "diabetes.csv", row.names = FALSE)
>
> # Step 2: Loading the CSV dataset
> diabetes_data <- read.csv("diabetes.csv")
>
> # Step 3: View the first few rows of the dataset
> head(diabetes_data)
  Age BloodPressure Glucose Insulin  BMI DiabetesPedigreeFunction Outcome
1  25            80     110      85 22.5                     0.20       0
2  30            85     120      90 23.0                     0.30       1
3  35            88     140     100 24.5                     0.40       0
4  40            90     160     120 26.0                     0.50       1
5  45            95     170     130 28.5                     0.45       1
6  50           100     180     140 30.0                     0.35       0
>
>
> # Step 2: View the structure of the dataset to understand it
> str(diabetes_data)
'data.frame':   12 obs. of  7 variables:
 $ Age                     : int  25 30 35 40 45 50 55 60 65 70 ...
 $ BloodPressure           : int  80 85 88 90 95 100 105 110 115 120 ...
 $ Glucose                 : int  110 120 140 160 170 180 190 200 210 220 ...
 $ Insulin                 : int  85 90 100 120 130 140 150 160 170 180 ...
 $ BMI                     : num  22.5 23 24.5 26 28.5 30 32.5 34 36 38 ...
 $ DiabetesPedigreeFunction: num  0.2 0.3 0.4 0.5 0.45 0.35 0.55 0.65 0.75 0.85 ...
 $ Outcome                 : int  0 1 0 1 1 0 1 1 0 1 ...
>
> # Step 3: Creating Age groups (e.g., <30, 30-40, 40-50, 50-60, >60)
> diabetes_data$AgeGroup <- cut(diabetes_data$Age, breaks = c(-Inf, 30, 40, 50, 60, Inf),
```

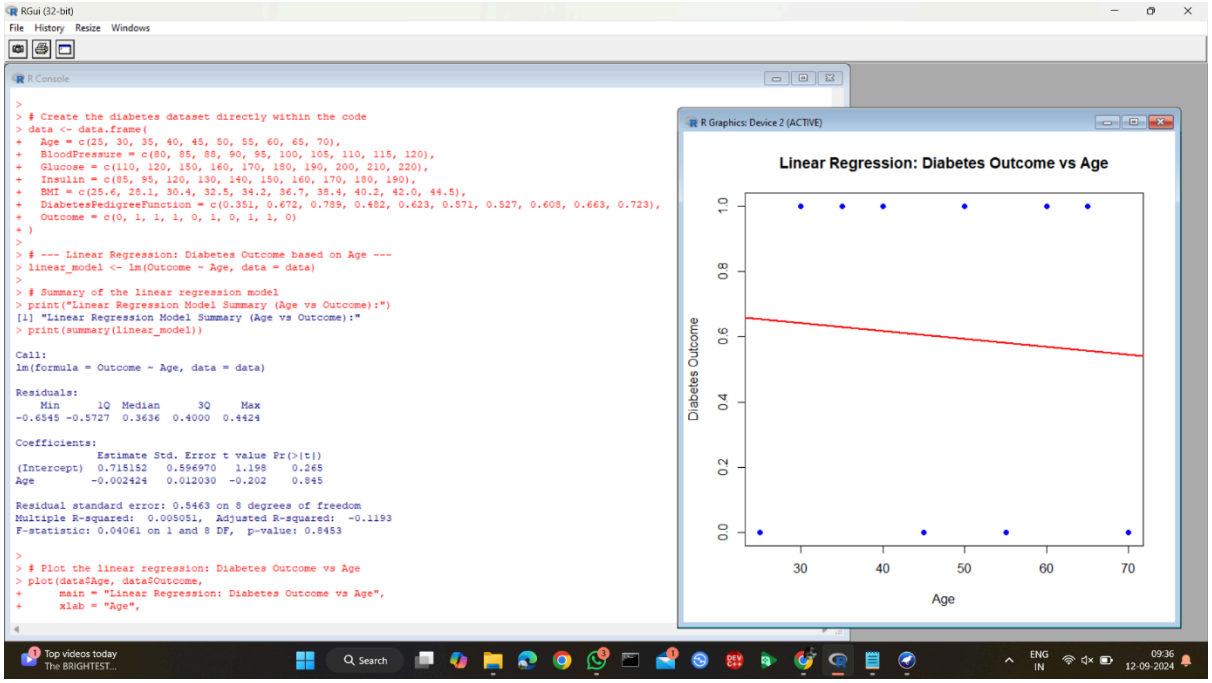



```
2  30            85     120      90 23.0                     0.30       1
3  35            88     140     100 24.5                     0.40       0
4  40            90     160     120 26.0                     0.50       1
5  45            95     170     130 28.5                     0.45       1
6  50           100     180     140 30.0                     0.35       0
>
>
> # Step 2: View the structure of the dataset to understand it
> str(diabetes_data)
'data.frame':   12 obs. of  7 variables:
 $ Age                     : int  25 30 35 40 45 50 55 60 65 70 ...
 $ BloodPressure           : int  80 85 88 90 95 100 105 110 115 120 ...
 $ Glucose                 : int  110 120 140 160 170 180 190 200 210 220 ...
 $ Insulin                 : int  85 90 100 120 130 140 150 160 170 180 ...
 $ BMI                     : num  22.5 23 24.5 26 28.5 30 32.5 34 36 38 ...
 $ DiabetesPedigreeFunction: num  0.2 0.3 0.4 0.5 0.45 0.35 0.55 0.65 0.75 0.85 ...
 $ Outcome                 : int  0 1 0 1 1 0 1 1 0 1 ...
>
> # Step 3: Creating Age groups (e.g., <30, 30-40, 40-50, 50-60, >60)
> diabetes_data$AgeGroup <- cut(diabetes_data$Age, breaks = c(-Inf, 30, 40, 50, 60, Inf),
+                               labels = c("<30", "30-40", "40-50", "50-60", ">60"))
>
> # Step 4: Scatter plot: Blood Pressure vs Age using base R
> plot(diabetes_data$Age, diabetes_data$BloodPressure,
+      main = "Scatter Plot: Blood Pressure vs Age",
+      xlab = "Age",
+      ylab = "Blood Pressure",
+      col = "blue",
+      pch = 19)
>
> # Step 5: Calculate average blood pressure for each age group
> bp_by_agegroup <- aggregate(BloodPressure ~ AgeGroup, data = diabetes_data, FUN = mean)
>
> # Step 6: Bar chart: Average Blood Pressure by Age Group using base R
> barplot(bp_by_agegroup$BloodPressure,
+         names.arg = bp_by_agegroup$AgeGroup,
+         col = "lightblue",
+         main = "Bar Chart: Average Blood Pressure by Age Group",
+         xlab = "Age Group",
+         ylab = "Average Blood Pressure")
>
```

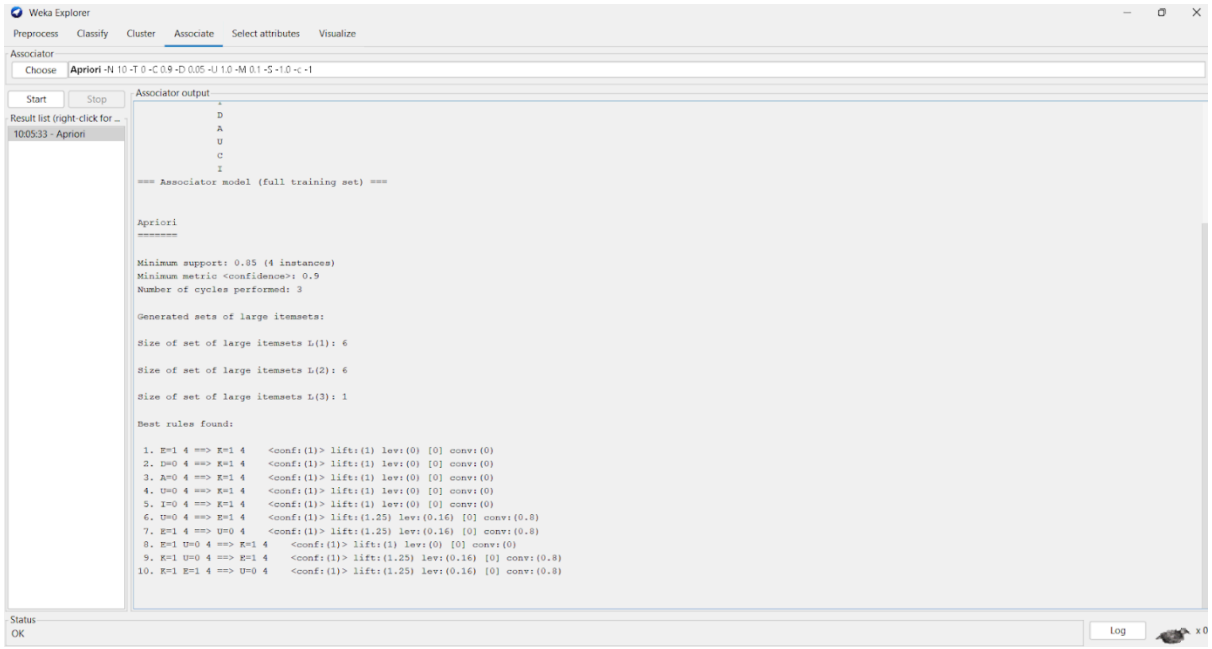


6.

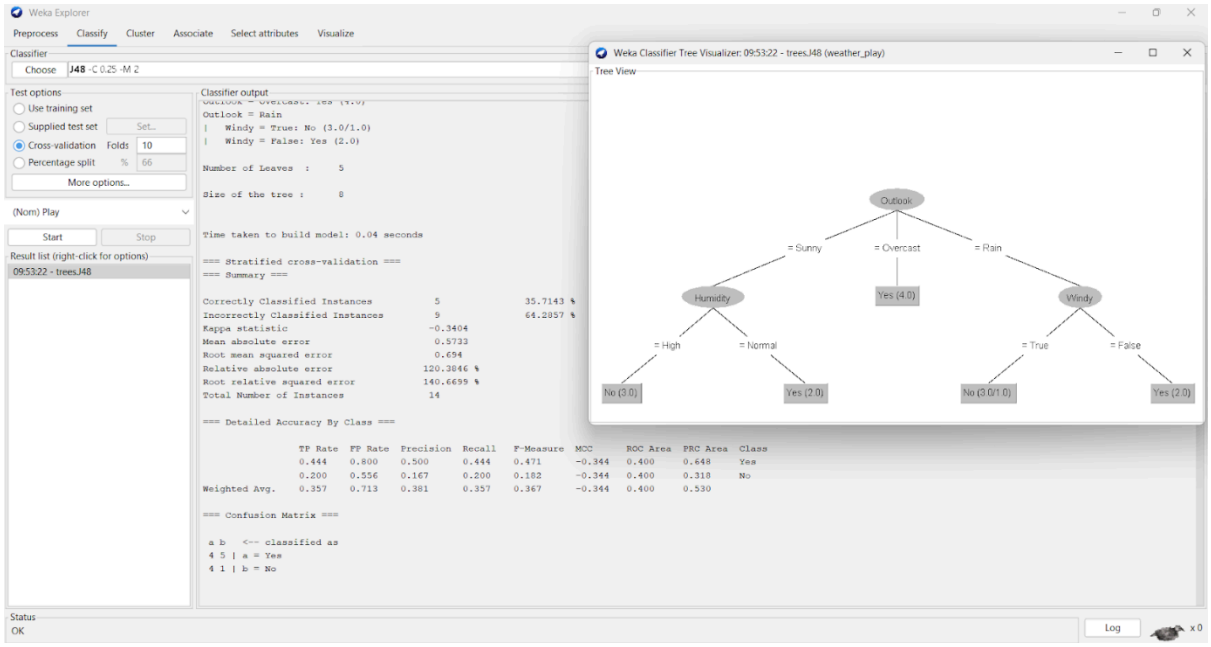

RGui (32-bit)
File History Resize Windows
R Console
R Graphics: Device 2 (ACTIVE)
Multiple Regression Coefficients
Coefficient Estimate
1.5
1.0
0.5
0.0
-0.5
(Intercept)
Age
Glucose
BMI
Variables

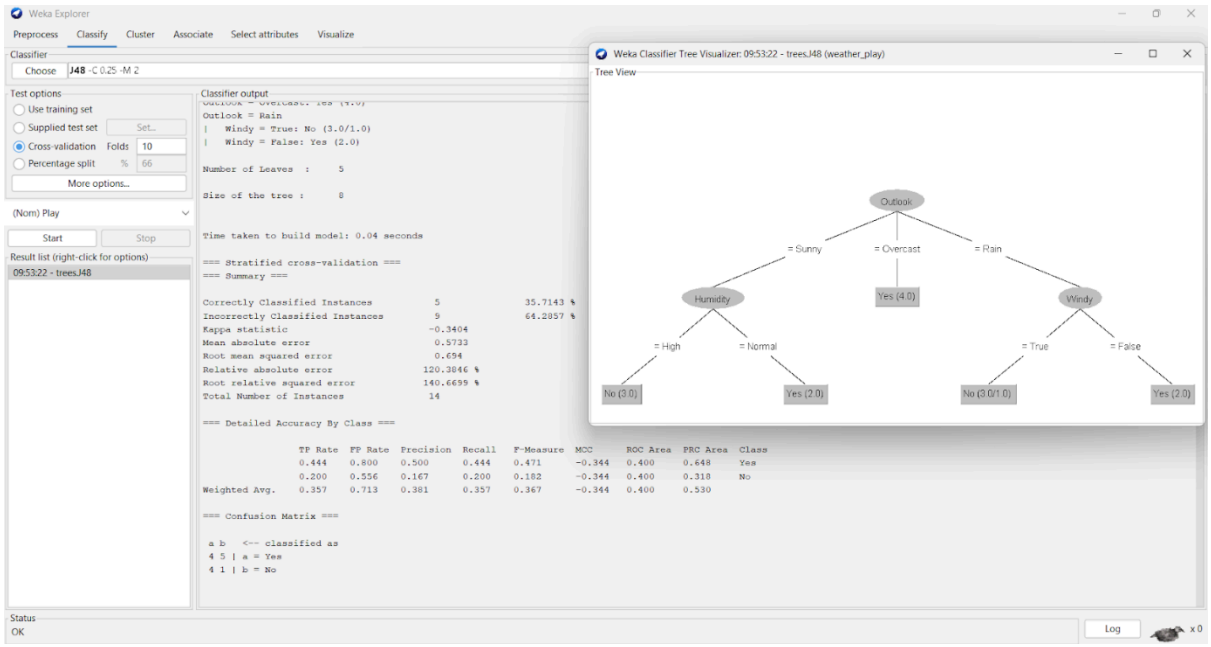

RGui (32-bit)
File History Resize Windows
R Console
R Graphics: Device 2 (ACTIVE)
Linear Regression: Diabetes Outcome vs Age
Diabetes Outcome
1.0
0.8
0.6
0.4
0.2
0.0
30
40
50
60
70
Age

7.

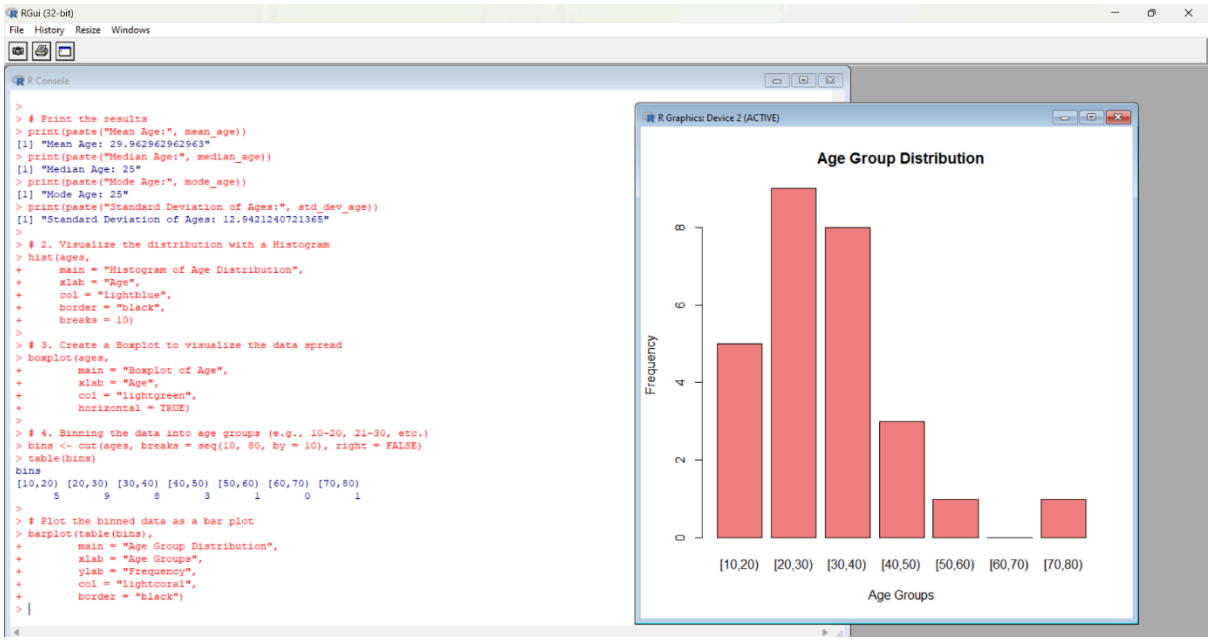

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Start
Stop
Result list (right-click for ...
10:05:33 - Apriori
Associator output
D
A
U
C
I
=== Associator model (full training set) ===
Apriori
=======
Minimum support: 0.85 (4 instances)
Minimum metric <confidence>: 0.9
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=1 4 ==> K=1 4 <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
2. D=0 4 ==> K=1 4 <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
3. A=0 4 ==> K=1 4 <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
4. U=0 4 ==> K=1 4 <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
5. I=0 4 ==> K=1 4 <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
6. U=0 4 ==> E=1 4 <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
7. E=1 4 ==> U=0 4 <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
8. E=1 U=0 4 ==> K=1 4 <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
9. K=1 U=0 4 ==> E=1 4 <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
10. K=1 E=1 4 ==> U=0 4 <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
Status
OK
Log
x 0

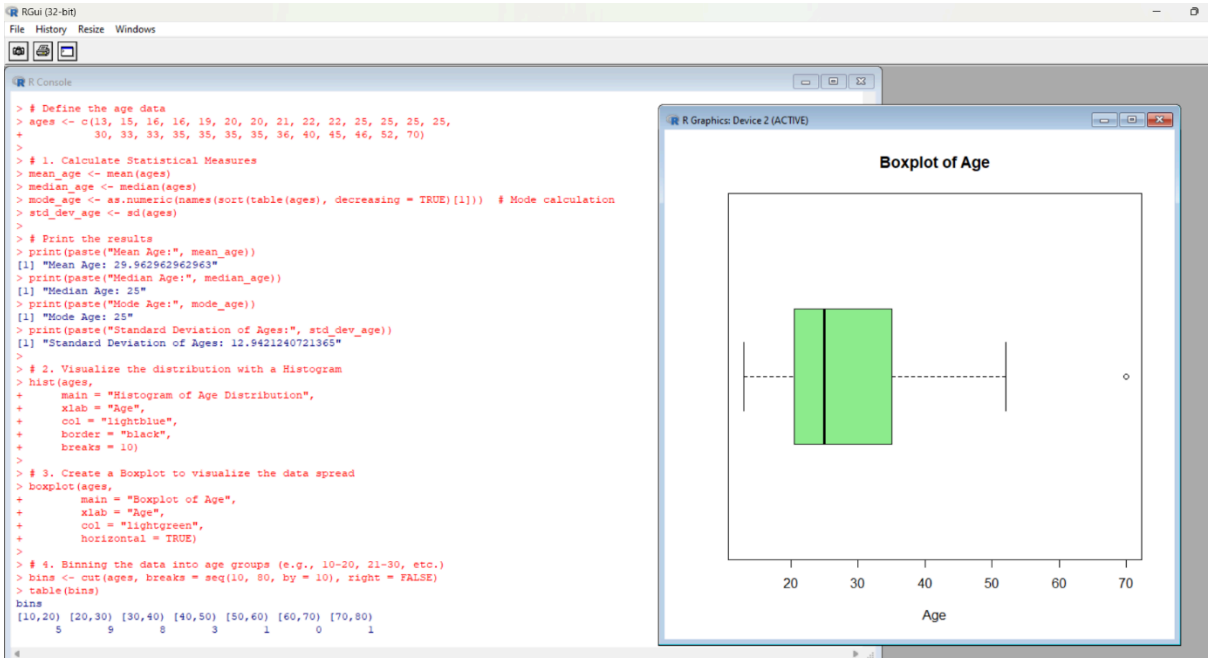
8.


Weka Explorer
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) Play
Start
Stop
Result list (right-click for options)
09:53:22 - trees.J48
Classifier output
Outlook = Rain
| Windy = True: No (3.0/1.0)
| Windy = False: Yes (2.0)
Number of Leaves : 5
Size of the tree : 8
Time taken to build model: 0.04 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 5 35.7143 %
Incorrectly Classified Instances 9 64.2857 %
Kappa statistic -0.3404
Mean absolute error 0.5733
Root mean squared error 0.694
Relative absolute error 120.3846 %
Root relative squared error 140.6699 %
Total Number of Instances 14
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.444 0.800 0.500 0.444 0.471 -0.344 0.400 0.648 Yes
0.200 0.556 0.167 0.200 0.182 -0.344 0.400 0.318 No
Weighted Avg. 0.357 0.713 0.381 0.357 0.367 -0.344 0.400 0.530
=== Confusion Matrix ===
a b <-- classified as
4 5 | a = Yes
4 1 | b = No
Status
OK
Log
x 0
Weka Classifier Tree Visualizer: 09:53:22 - trees.J48 (weather_play)
Tree View
Outlook
= Sunny
= Overcast
= Rain
Humidity
Yes (4.0)
Windy
= High
= Normal
= True
= False
No (3.0)
Yes (2.0)
No (3.0/1.0)
Yes (2.0)

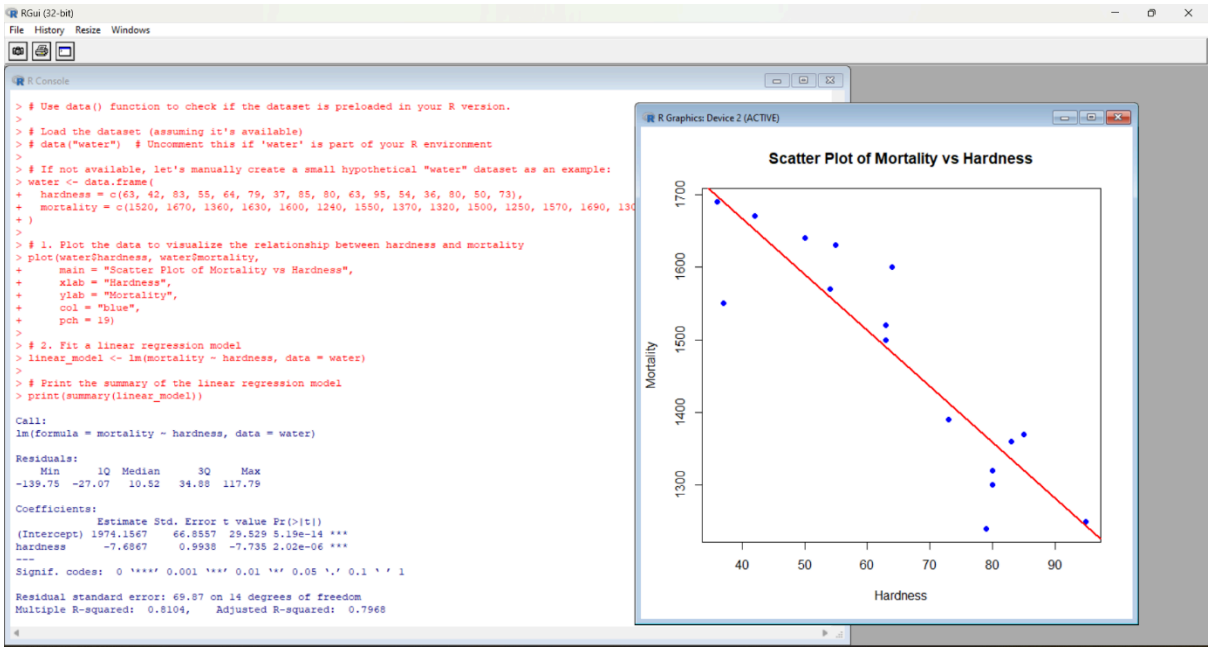

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) Play
Start
Stop
Result list (right-click for options)
09:53:22 - trees.J48
Classifier output
Outlook = Rain
| Windy = True: No (3.0/1.0)
| Windy = False: Yes (2.0)
Number of Leaves : 5
Size of the tree : 8
Time taken to build model: 0.04 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 5 35.7143 %
Incorrectly Classified Instances 9 64.2857 %
Kappa statistic -0.3404
Mean absolute error 0.5733
Root mean squared error 0.694
Relative absolute error 120.3846 %
Root relative squared error 140.6699 %
Total Number of Instances 14
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.444 0.800 0.500 0.444 0.471 -0.344 0.400 0.648 Yes
0.200 0.556 0.167 0.200 0.182 -0.344 0.400 0.318 No
Weighted Avg. 0.357 0.713 0.381 0.357 0.367 -0.344 0.400 0.530
=== Confusion Matrix ===
a b <-- classified as
4 5 | a = Yes
4 1 | b = No
Status
OK
Log
x 0
Weka Classifier Tree Visualizer: 09:53:22 - trees.J48 (weather_play)
Tree View
Outlook
= Sunny
= Overcast
= Rain
Humidity
Yes (4.0)
Windy
= High
= Normal
= True
= False
No (3.0)
Yes (2.0)
No (3.0/1.0)
Yes (2.0)

9.


RGui (32-bit)
File History Resize Windows
R Console
> # Print the results
> print(paste("Mean Age:", mean_age))
[1] "Mean Age: 29.962962962963"
> print(paste("Median Age:", median_age))
[1] "Median Age: 25"
> print(paste("Mode Age:", mode_age))
[1] "Mode Age: 25"
> print(paste("Standard Deviation of Ages:", std_dev_age))
[1] "Standard Deviation of Ages: 12.9421240721365"
> # 2. Visualize the distribution with a Histogram
> hist(ages,
+ main = "Histogram of Age Distribution",
+ xlab = "Age",
+ col = "lightblue",
+ border = "black",
+ breaks = 10)
> # 3. Create a Boxplot to visualize the data spread
> boxplot(ages,
+ main = "Boxplot of Age",
+ xlab = "Age",
+ col = "lightgreen",
+ horizontal = TRUE)
> # 4. Binning the data into age groups (e.g., 10-20, 21-30, etc.)
> bins <- cut(ages, breaks = seq(10, 80, by = 10), right = FALSE)
> table(bins)
bins
[10,20) [20,30) [30,40) [40,50) [50,60) [60,70) [70,80)
5 9 8 3 1 0 1
> # Plot the binned data as a bar plot
> barplot(table(bins),
+ main = "Age Group Distribution",
+ xlab = "Age Groups",
+ ylab = "Frequency",
+ col = "lightcoral",
+ border = "black")
R Graphics: Device 2 (ACTIVE)
Age Group Distribution
Frequency
0
2
4
6
8
[10,20)
[20,30)
[30,40)
[40,50)
[50,60)
[60,70)
[70,80)
Age Groups

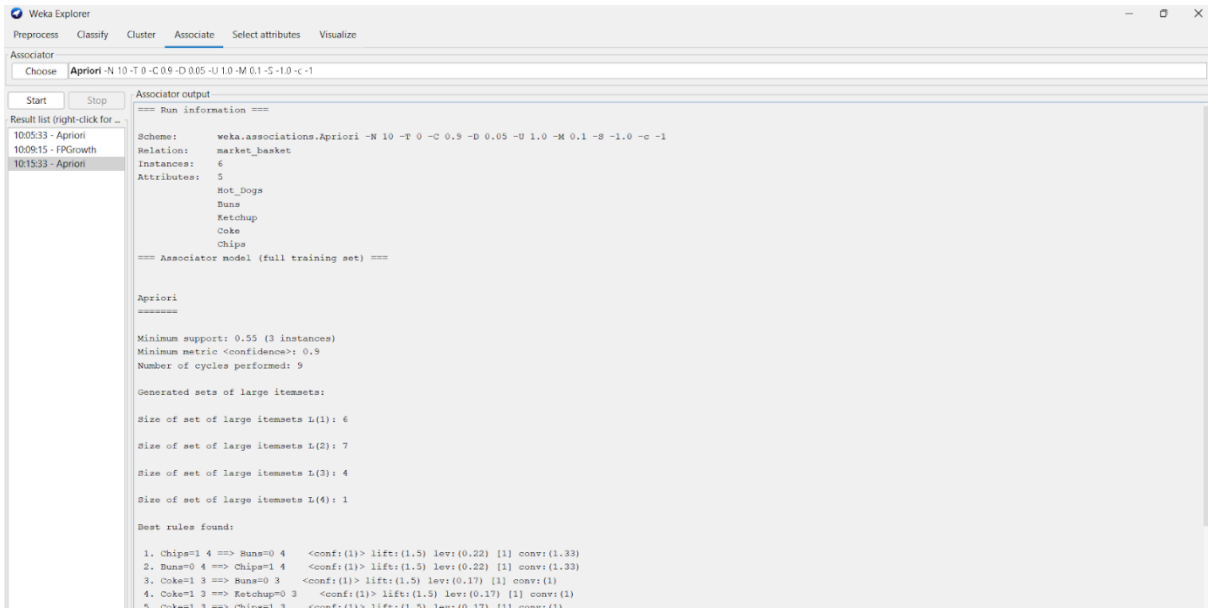

RGui (32-bit)
File History Resize Windows
R Console
> # Define the age data
> ages <- c(13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25,
+ 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70)
> # 1. Calculate Statistical Measures
> mean_age <- mean(ages)
> median_age <- median(ages)
> mode_age <- as.numeric(names(sort(table(ages), decreasing = TRUE)[1])) # Mode calculation
> std_dev_age <- sd(ages)
> # Print the results
> print(paste("Mean Age:", mean_age))
[1] "Mean Age: 29.962962962963"
> print(paste("Median Age:", median_age))
[1] "Median Age: 25"
> print(paste("Mode Age:", mode_age))
[1] "Mode Age: 25"
> print(paste("Standard Deviation of Ages:", std_dev_age))
[1] "Standard Deviation of Ages: 12.9421240721365"
> # 2. Visualize the distribution with a Histogram
> hist(ages,
+ main = "Histogram of Age Distribution",
+ xlab = "Age",
+ col = "lightblue",
+ border = "black",
+ breaks = 10)
> # 3. Create a Boxplot to visualize the data spread
> boxplot(ages,
+ main = "Boxplot of Age",
+ xlab = "Age",
+ col = "lightgreen",
+ horizontal = TRUE)
> # 4. Binning the data into age groups (e.g., 10-20, 21-30, etc.)
> bins <- cut(ages, breaks = seq(10, 80, by = 10), right = FALSE)
> table(bins)
bins
[10,20) [20,30) [30,40) [40,50) [50,60) [60,70) [70,80)
5 9 8 3 1 0 1
R Graphics: Device 2 (ACTIVE)
Boxplot of Age
20
30
40
50
60
70
Age

10.


RGui (32-bit)
File History Resize Windows
R Console
> # Use data() function to check if the dataset is preloaded in your R version.
> # Load the dataset (assuming it's available)
> # data("water") # Uncomment this if 'water' is part of your R environment
> # If not available, let's manually create a small hypothetical "water" dataset as an example:
> water <- data.frame(
+ hardness = c(63, 42, 83, 55, 64, 79, 37, 85, 80, 63, 95, 54, 36, 80, 50, 73),
+ mortality = c(1520, 1670, 1360, 1630, 1600, 1240, 1550, 1370, 1320, 1500, 1250, 1570, 1690, 13
+)
> # 1. Plot the data to visualize the relationship between hardness and mortality
> plot(water$hardness, water$mortality,
+ main = "Scatter Plot of Mortality vs Hardness",
+ xlab = "Hardness",
+ ylab = "Mortality",
+ col = "blue",
+ pch = 19)
> # 2. Fit a linear regression model
> linear_model <- lm(mortality ~ hardness, data = water)
> # Print the summary of the linear regression model
> print(summary(linear_model))
Call:
lm(formula = mortality ~ hardness, data = water)
Residuals:
Min 1Q Median 3Q Max
-139.75 -27.07 10.52 34.88 117.79
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) 1974.1567 66.8557 29.529 5.19e-14 ***
hardness -7.6867 0.9938 -7.735 2.02e-06 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 69.87 on 14 degrees of freedom
Multiple R-squared: 0.8104, Adjusted R-squared: 0.7968
R Graphics: Device 2 (ACTIVE)
Scatter Plot of Mortality vs Hardness
Mortality
Hardness
1300
1400
1500
1600
1700
40
50
60
70
80
90

11.


Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Associator
Choose Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Start Stop
Result list (right-click for ...
10:05:33 - Apriori
10:09:15 - FPGrowth
10:15:33 - Apriori
Associator output
=== Run information ===
Scheme: weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation: market_basket
Instances: 6
Attributes: 5
Hot_Dogs
Buns
Ketchup
Coke
Chips
=== Associator model (full training set) ===
Apriori
=======
Minimum support: 0.55 (3 instances)
Minimum metric <confidence>: 0.9
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=1 4 ==> Buns=0 4 <conf:(1)> lift:(1.5) lev:(0.22) [1] conv:(1.33)
2. Buns=0 4 ==> Chips=1 4 <conf:(1)> lift:(1.5) lev:(0.22) [1] conv:(1.33)
3. Coke=1 3 ==> Buns=0 3 <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)
4. Coke=1 3 ==> Ketchup=0 3 <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)