

## 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:

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

```
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:

```
> #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)
[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
> |
```

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:

```
> 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
> 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
```

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

```
mean(age)
```

```
median(age)
```

```
sd(age)
```

```
mean(fat)
```

```
median(fat)
```

```
sd(fat)
```

```
#boxplot
```

```
boxplot(age,fat)
```

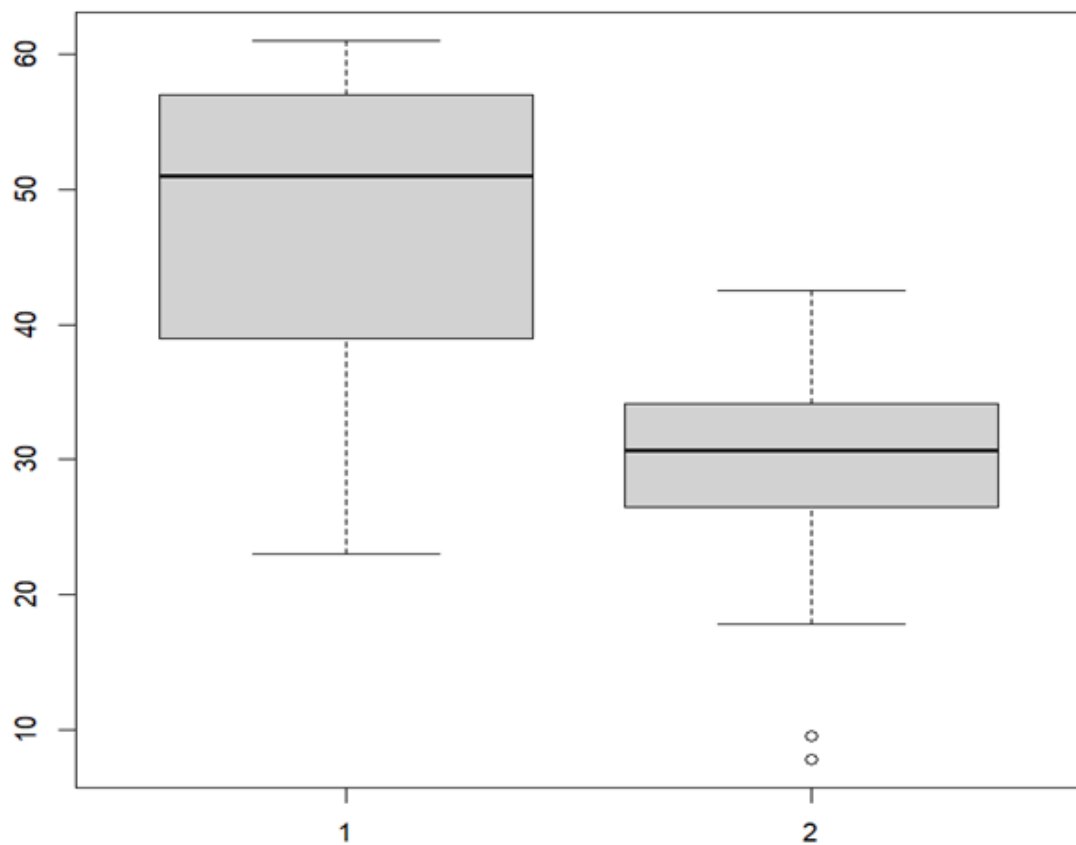
```
#scatter plot
```

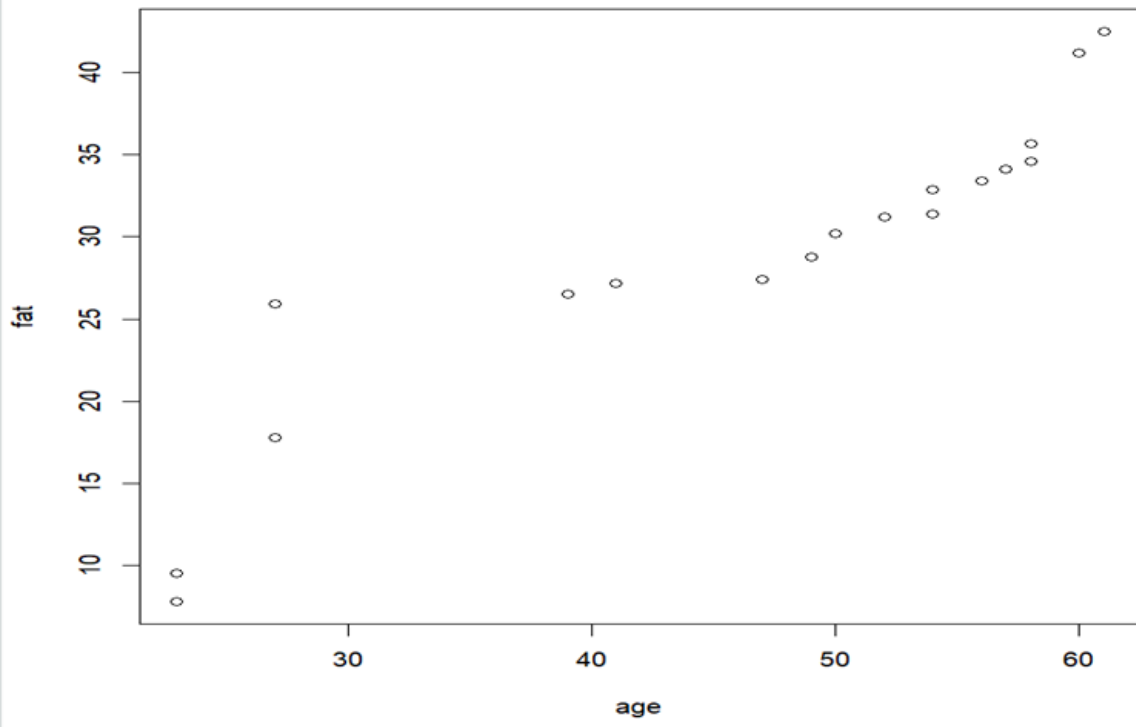
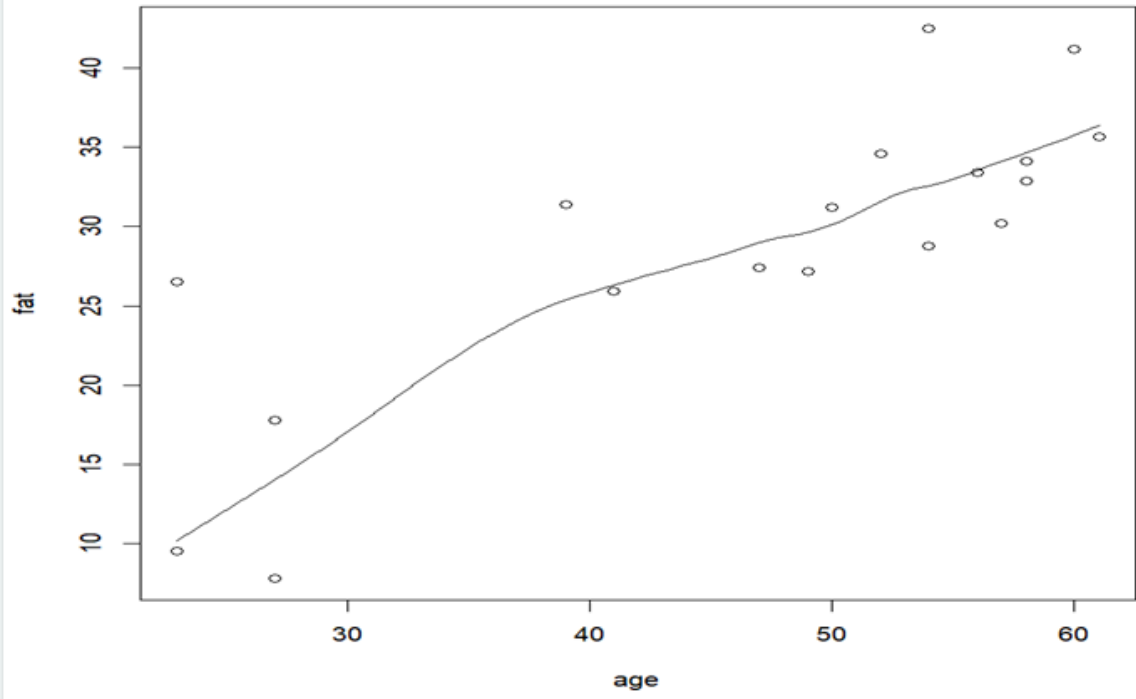
```
scatter.smooth(age,fat)
```

```
#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:**

```

print(min_max)
.] 0.3157895
#z-score
m=mean(v)
s<-12.94
z_score=(35-m)/s
print(z_score)
.] -0.8844238
#decimal scaling
m<-35
j=max(m)<1
decimal_scaling=m/10^j
print(decimal_scaling)
.] 35
|

```

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:



```

pencils<-c(9,25,23,12,11,6,7,8,9,10)
mean(pencils)
.] 12
median(pencils)
.] 9.5
mode=names(table(pencils))[table(pencils)==max(table(pencils))]
mode
.] "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:

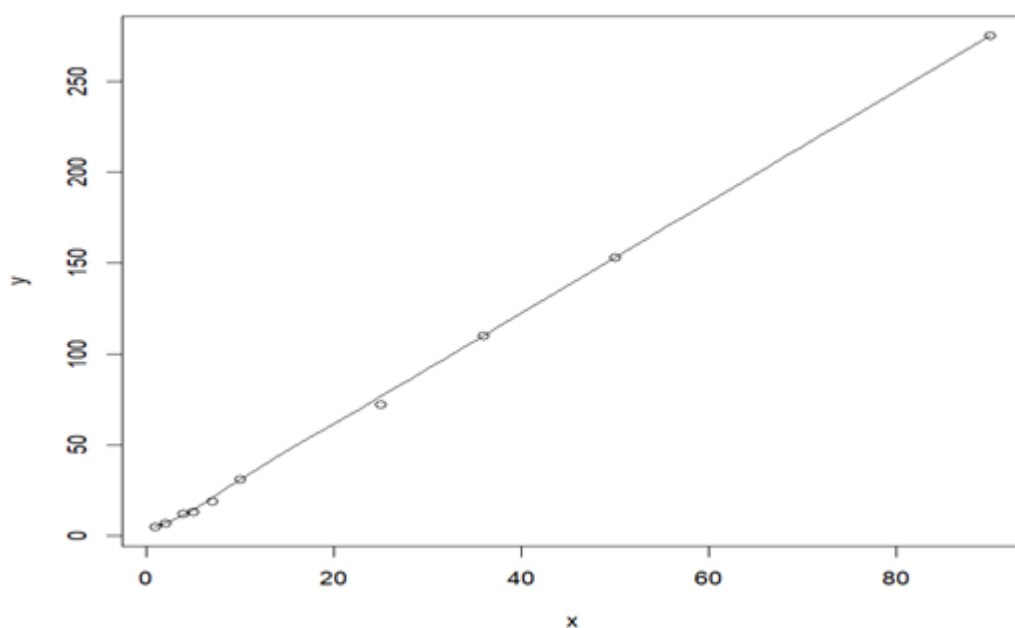
```
#scatterplot
```

```
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:")

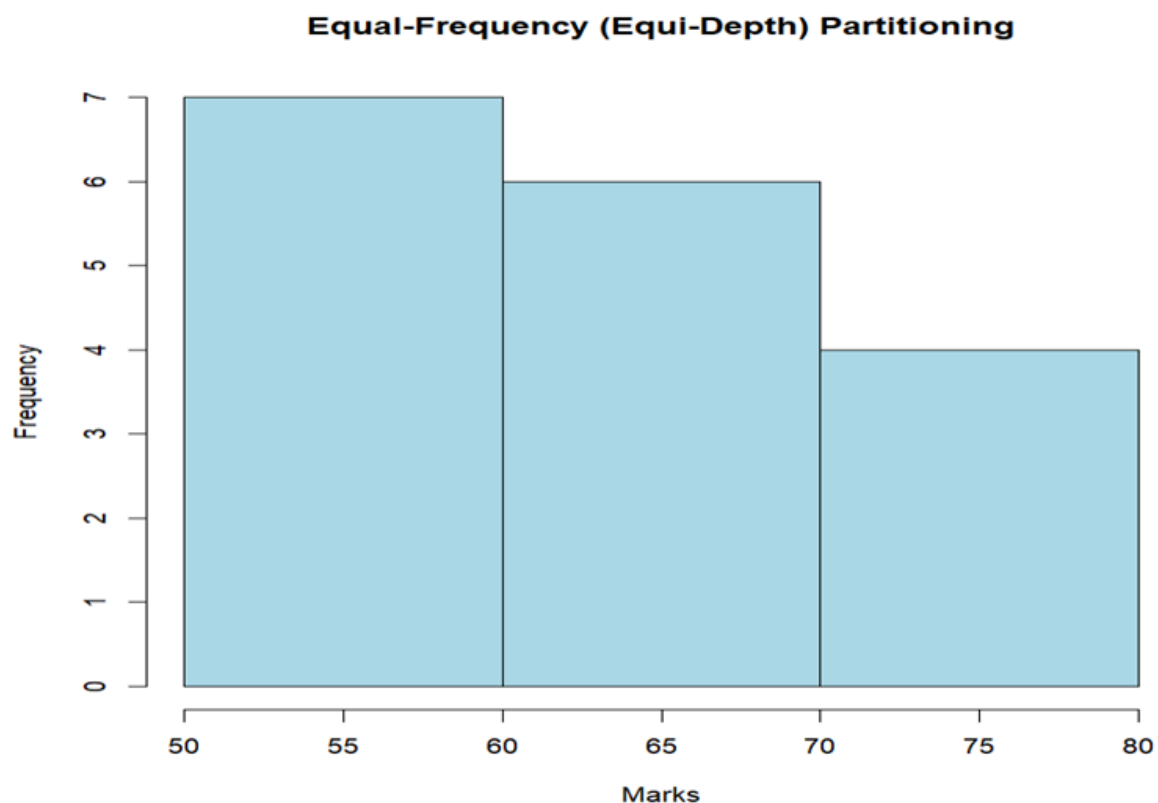
print(smoothed_data_by_boundaries)
```

## Output:

```

> print(smoothed_data_by_mean)
[1] 18.9375 18.9375 18.9375 18.9375 18.9375 18.9375 18.9375 18.9375
[9] 18.9375 18.9375 18.9375 18.9375 18.9375 18.9375 18.9375 18.9375
[17] 43.7500 43.7500 43.7500 43.7500 72.7500 72.7500 72.7500 72.7500
> print("Smoothed data by bin median:")
[1] "Smoothed data by bin median:"
> print(smoothed_data_by_median)
[1] 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0
[14] 20.0 20.0 20.0 45.0 45.0 45.0 45.0 72.5 72.5 72.5 72.5
> print("Smoothed data by bin boundaries:")
[1] "Smoothed data by bin boundaries:"
> print(smoothed_data_by_boundaries)
(10.9,32.3]1 (10.9,32.3]2 (10.9,32.3]1 (10.9,32.3]2 (10.9,32.3]1
      11      30      11      30      11
(10.9,32.3]2 (10.9,32.3]1 (10.9,32.3]2 (10.9,32.3]1 (10.9,32.3]2
      30      11      30      11      30
(10.9,32.3]1 (10.9,32.3]2 (10.9,32.3]1 (10.9,32.3]2 (10.9,32.3]1
      11      30      11      30      11
(10.9,32.3]2 (10.9,32.3]1 (10.9,32.3]2 (10.9,32.3]1 (10.9,32.3]2
      30      11      30      11      30

```



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:

```
v<-c(78.3,81.8,82,74.2,83.4,84.5,82.9,77.5,80.9,70.6)
IQR(v)
.] 4.975
sd(v)
.] 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, 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:

```
> #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)
25%
20.5
> quantile(age,.75)
75%
35
```

# Day 2

## 12. 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:

```
data <- data.frame(
  Age = rep(c("5-6 years", "7-8 years", "9-10 years"), each = 3),
  A = c(18, 2, 20, 22, 28, 10, 20, 40, 40),
  B = c(22, 28, 10, 20, 40, 40, 30, 45, 50),
  C = c(20, 40, 40, 30, 45, 50, 15, 35, 25)
)

covariance_BC <- cov(data$B, data$C)

cat("Covariance between B and C:", covariance_BC, "\n")

covariance_matrix <- cov(data[, c("A", "B", "C")])

cat("Covariance matrix:\n", covariance_matrix, "\n")
```

```
correlation_BC <- cor(data$B, data$C)

cat("Correlation between B and C:", correlation_BC, "\n")

correlation_matrix <- cor(data[, c("A", "B", "C")])

cat("Correlation matrix:\n", correlation_matrix, "\n")
```

## **output:**

```
> data <- data.frame(

+ Age = rep(c("5-6 years", "7-8 years", "9-10 years"), each = 3),

+ A = c(18, 2, 20, 22, 28, 10, 20, 40, 40),

+ B = c(22, 28, 10, 20, 40, 40, 30, 45, 50),

+ C = c(20, 40, 40, 30, 45, 50, 15, 35, 25)

+ )

> covariance_BC <- cov(data$B, data$C)

> cat("Covariance between B and C:", covariance_BC, "\n")

Covariance between B and C: 16.875

> covariance_matrix <- cov(data[, c("A", "B", "C")])

> cat("Covariance matrix:\n", covariance_matrix, "\n")

Covariance matrix:

156.4444 84.83333 -38.33333 84.83333 171 16.875 -38.33333 16.875 137.5

> correlation_BC <- cor(data$B, data$C)

> cat("Correlation between B and C:", correlation_BC, "\n")

Correlation between B and C: 0.1100511

> correlation_matrix <- cor(data[, c("A", "B", "C")])

> cat("Correlation matrix:\n", correlation_matrix, "\n")

Correlation matrix:

1 0.5186667 -0.2613636 0.5186667 1 0.1100511 -0.2613636 0.1100511 1
```

13. 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, 18, 18, 18, 20, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 25, 25, 25, 25, 25, 28, 28, 30, 30, 30)

## Input:

```
# Given data
```

```
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, 18, 18, 18, 18, 18, 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) Equal-frequency partitioning with bin equal to 3
```

```
equal_freq_bins <- cut(prices, breaks = 3, labels = FALSE)
```

```
cat("(i) Equal-frequency partitioning bins:\n", equal_freq_bins, "\n")
```

```
18, 18, 18, 20, 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

```
# (ii) Data smoothing using bin means and bin boundary
```

```
bin_means <- tapply(prices, equal_freq_bins, mean)
```

```
bin_boundaries <- unique(cut(prices, breaks = 3, labels = FALSE, include.lowest = TRUE))
```

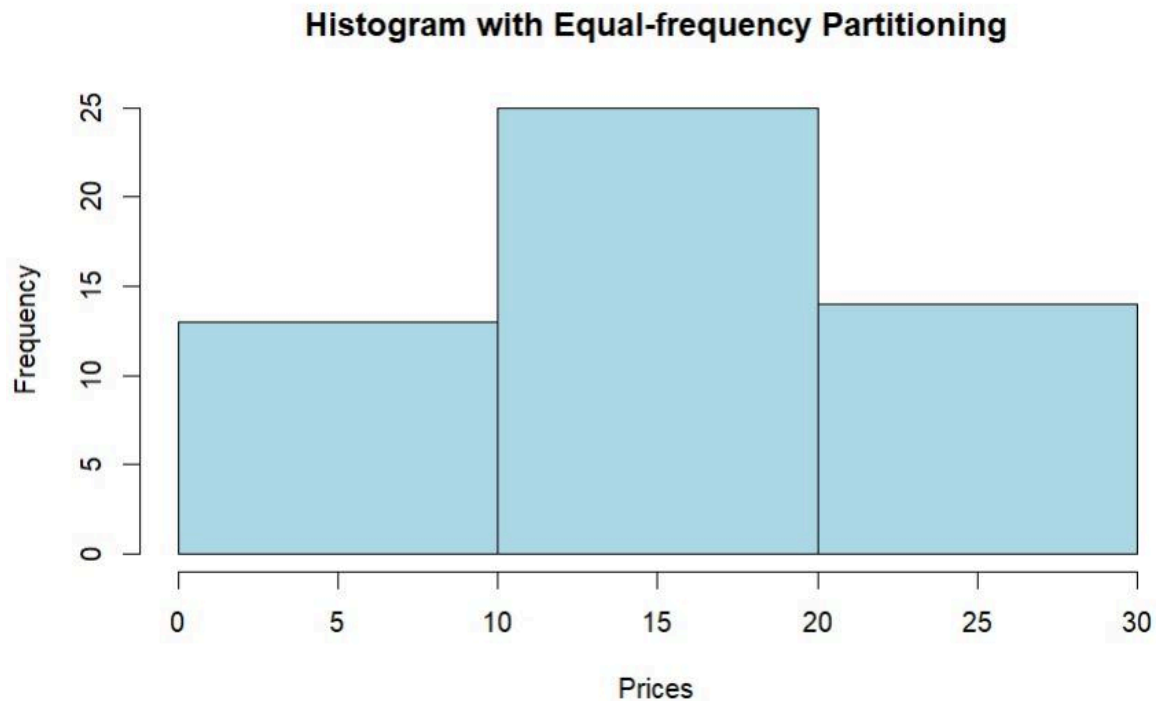
```
cat("(ii) Bin Means:\n", bin_means, "\n")
```

```
cat("Bin Boundaries:\n", bin_boundaries, "\n")
```

# (iii) Plot Histogram

```
hist(prices, breaks = 3, main = "Histogram with Equal-frequency Partitioning", xlab =  
"Prices", col = "lightblue", border = "black")
```

Output:



14. 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.

## INPUT:

```
classA <- c(76, 35, 47, 64, 95, 66, 89, 36, 84)
```

```
classB <- c(51, 56, 84, 60, 59, 70, 63, 66, 50)
```



```
meanA <- mean(classA)

meanB <- mean(classB)

medianA <- median(classA)

medianB <- median(classB)

rangeA <- range(classA)

rangeB <- range(classB)

cat("(i) Class A vs Class B:\n")

cat("Mean: Class A -", meanA, " Class B -", meanB, "\n")

cat("Median: Class A -", medianA, " Class B -", medianB, "\n")

cat("Range: Class A -", diff(rangeA), " Class B -", diff(rangeB), "\n")

boxplot(classA, classB, names = c("Class A", "Class B"), col = c("lightblue", "lightgreen"),
main = "Boxplot - Class A vs Class B", ylab = "Scores")

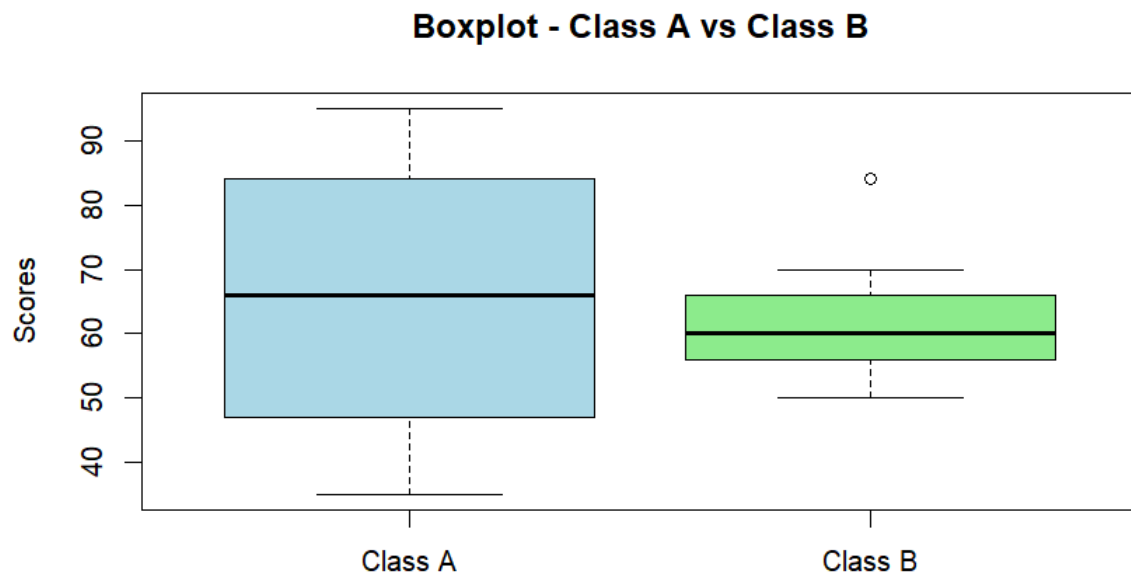
cat("(ii) Inferences from Boxplot:\n")

cat("- Class A has a wider range of scores compared to Class B.\n")

cat("- The median score for Class A is higher than that for Class B.\n")

cat("- Class A has an outlier which affects the mean.\n")
```

**output:**



15. 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_max_custom <- function(x, min_val, max_val) {
```

```
  return (x - min_val) / (max_val - min_val)
```

```
}
```

```
min_max_normalized_custom <- min_max_custom(data, 200, 1000)
```

```
min_max_normalized_default <- scale(data, center = min(data), scale = diff(range(data)))
```

```
z_score_normalized <- scale(data)
```

```
cat("Original Data: ", data, "\n\n")
```

```
cat("(a) Min-Max normalization with custom min and max values:\n")
```

```
cat("Normalized Data: ", min_max_normalized_custom, "\n\n")
```

```
cat("(b) Min-Max normalization with min = 0 and max = 1:\n")
```

```
cat("Normalized Data: ", min_max_normalized_default, "\n\n")
```

```
cat("(c) Z-score normalization:\n")
```

```
cat("Normalized Data: ", z_score_normalized, "\n")
```

## **output:**

Original Data: 200 300 400 600 1000

```
> cat("(a) Min-Max normalization with custom min and max values:\n")
```

(a) Min-Max normalization with custom min and max values:

```
> cat("Normalized Data: ", min_max_normalized_custom, "\n\n")
```

Normalized Data: 0 100 200 400 800

```
> cat("(b) Min-Max normalization with min = 0 and max = 1:\n")
```

(b) Min-Max normalization with min = 0 and max = 1:

```
> cat("Normalized Data: ", min_max_normalized_default, "\n\n")
```

Normalized Data: 0 0.125 0.25 0.5 1

```
> cat("(c) Z-score normalization:\n")
```

(c) Z-score normalization:

```
> cat("Normalized Data: ", z_score_normalized, "\n")
```

Normalized Data: -0.9486833 -0.6324555 -0.3162278 0.3162278 1.581139

16. 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:

```
data("AirPassengers")

start_value <- 100

bin_width <- 150

bin_breaks <- seq(start_value, 700, by = bin_width)

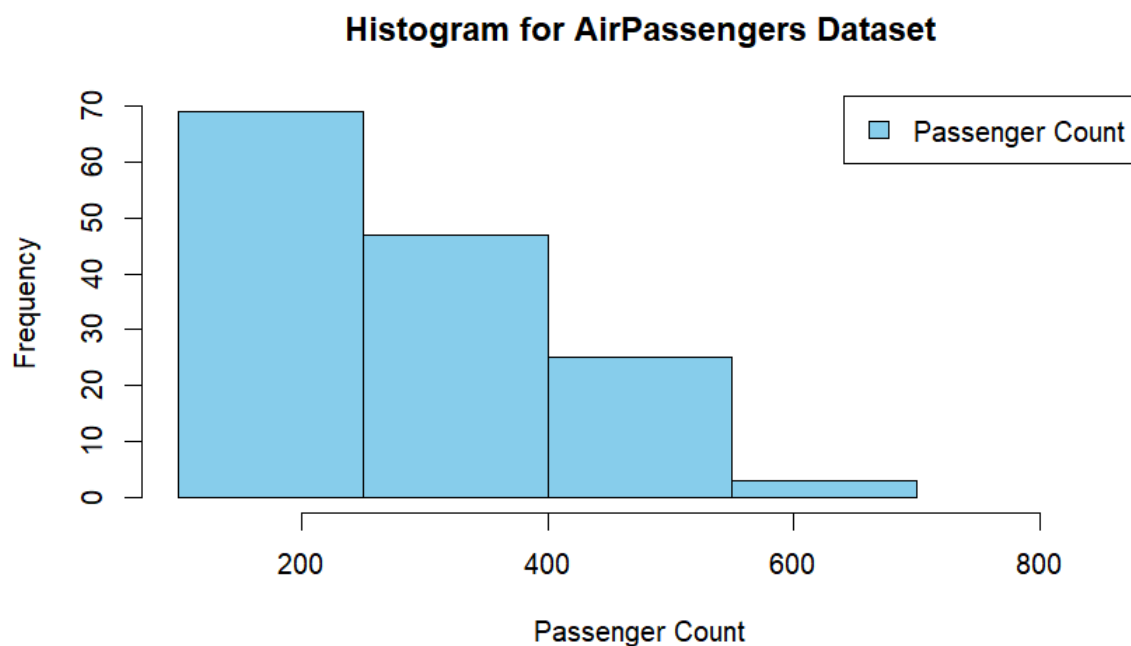
hist(AirPassengers, breaks = bin_breaks, xlim = c(start_value, max(bin_breaks) +
bin_width),

main = "Histogram for AirPassengers Dataset",

xlab = "Passenger Count", ylab = "Frequency", col = "skyblue", border = "black")

legend("topright", legend = c("Passenger Count"), fill = c("skyblue"))
```

## OUTPUT:



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

## INPUT

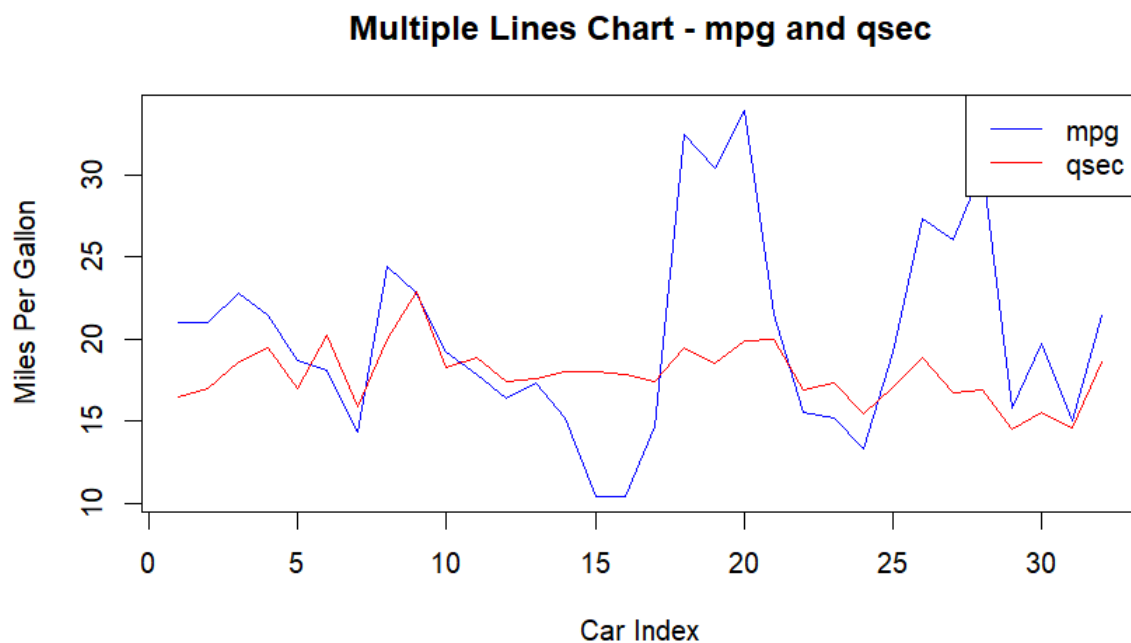
```
data(mtcars)
```

```
plot(mtcars$mpg, type = "l", col = "blue", xlab = "Car Index", ylab = "Miles Per Gallon", main = "Multiple Lines Chart - mpg and qsec")
```

```
lines(mtcars$qsec, col = "red")
```

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

## OUTPUT



18. 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(mtcars)

mortality <- mtcars$mpg

hardness <- mtcars$hp

plot(hardness, mortality, main = "Linear Regression: Mortality vs. Hardness",
     xlab = "Hardness", ylab = "Mortality", pch = 16, col = "blue")

linear_model <- lm(mortality ~ hardness)

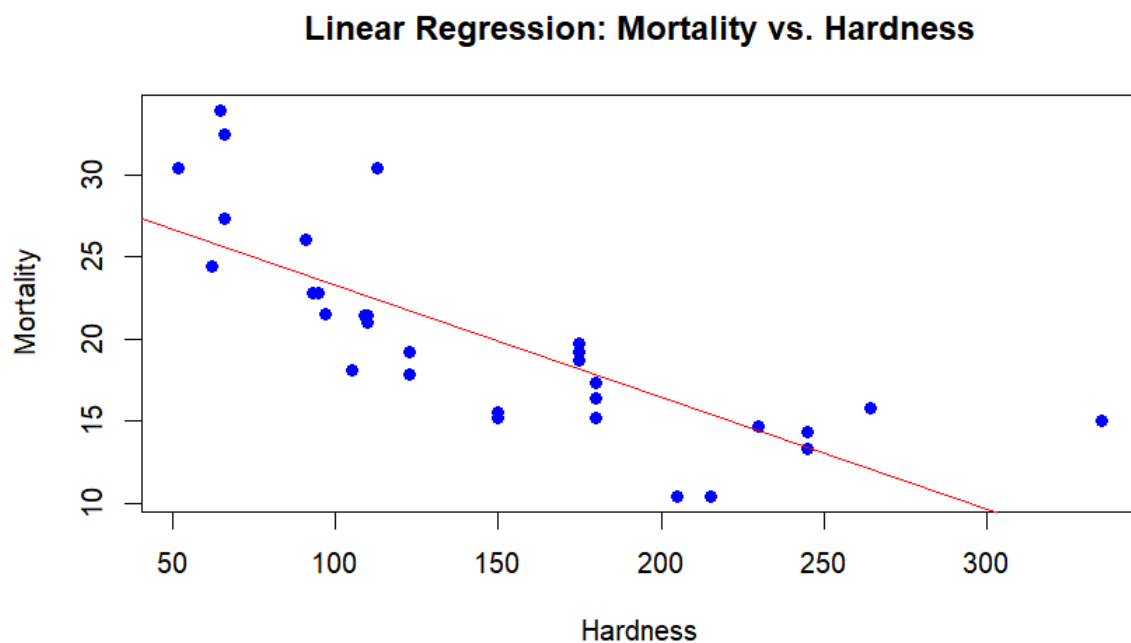
abline(linear_model, col = "red")

new_data <- data.frame(hardness = 88)

predicted_mortality <- predict(linear_model, newdata = new_data)

cat("Predicted Mortality for Hardness=88:", predicted_mortality, "\n")
```

Output:



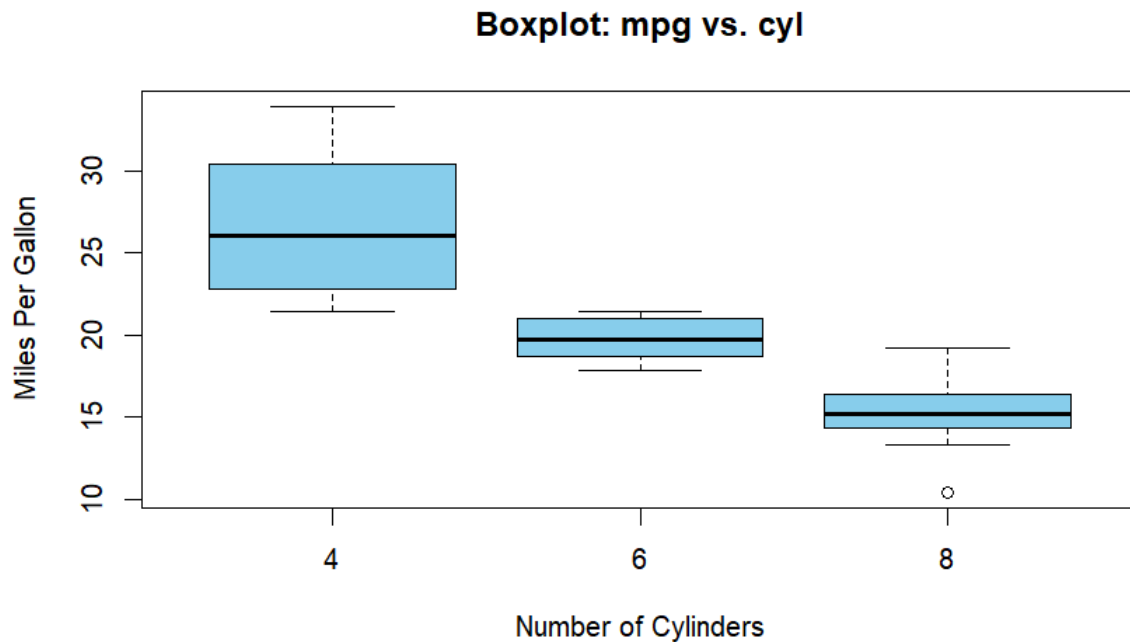
19. Create a Boxplot graph for the relation between "mpg"(miles per galloon) and "cyl"(number of Cylinders) for the dataset "mtcars" available in R Environment.

**Input:**

```
data(mtcars)

boxplot(mpg ~ cyl, data = mtcars, main = "Boxplot: mpg vs. cyl",
xlab = "Number of Cylinders", ylab = "Miles Per Gallon", col = "skyblue")
```

## OUTPUT



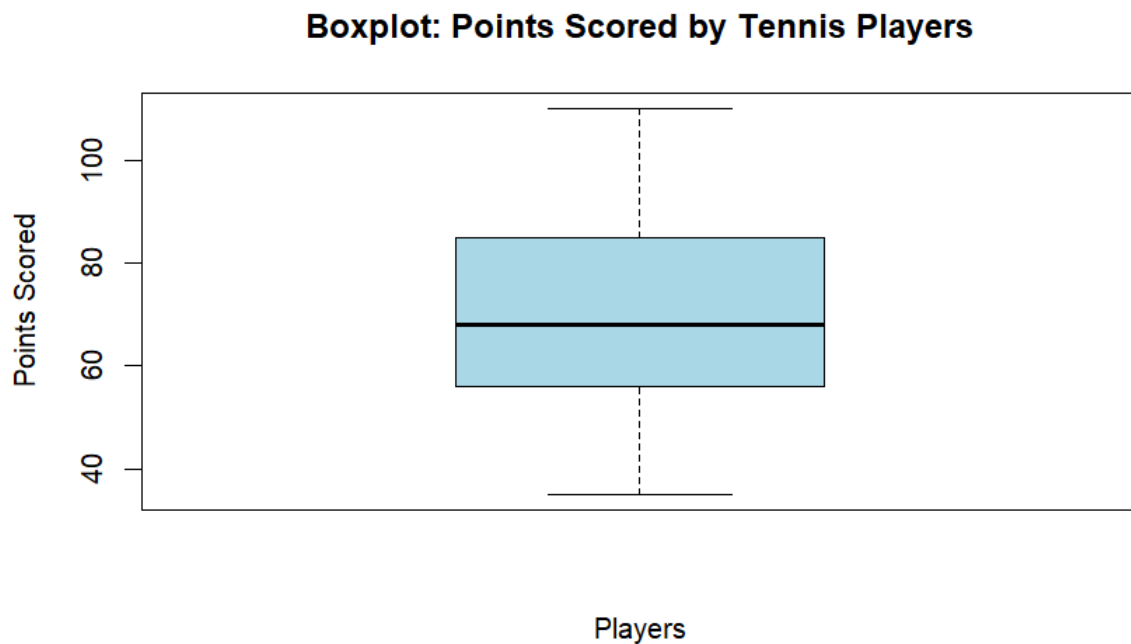
20. Assume the Tennis coach wants to determine if any of his team players are scoring outliers. To visualize the distribution of points scored by his players, then how can he decide to develop the box plot? Give suitable example using Boxplot visualization technique.

### Input:

```
points_scored <- c(35, 42, 48, 52, 56, 60, 62, 65, 68, 72, 76, 80, 85, 88, 92, 100, 110)

boxplot(points_scored, main = "Boxplot: Points Scored by Tennis Players",
xlab = "Players", ylab = "Points Scored", col = "lightblue", border = "black")
```

## 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:**

# Sample data (assuming the structure of your dataset)

```
data <- data.frame(
```

```
  Age = c(25, 30, 35, 40, 45, 50, 55, 60, 65, 70),
```

```
  BloodPressure = c(120, 130, 140, 150, 135, 145, 155, 160, 150, 140)
```

```
)
```

# Scatterplot

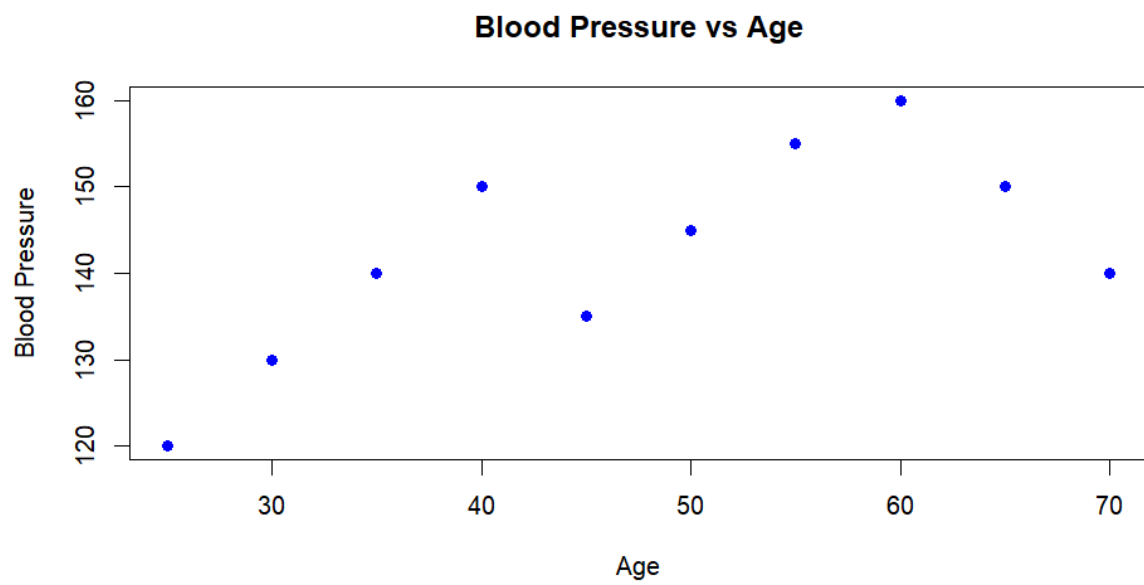
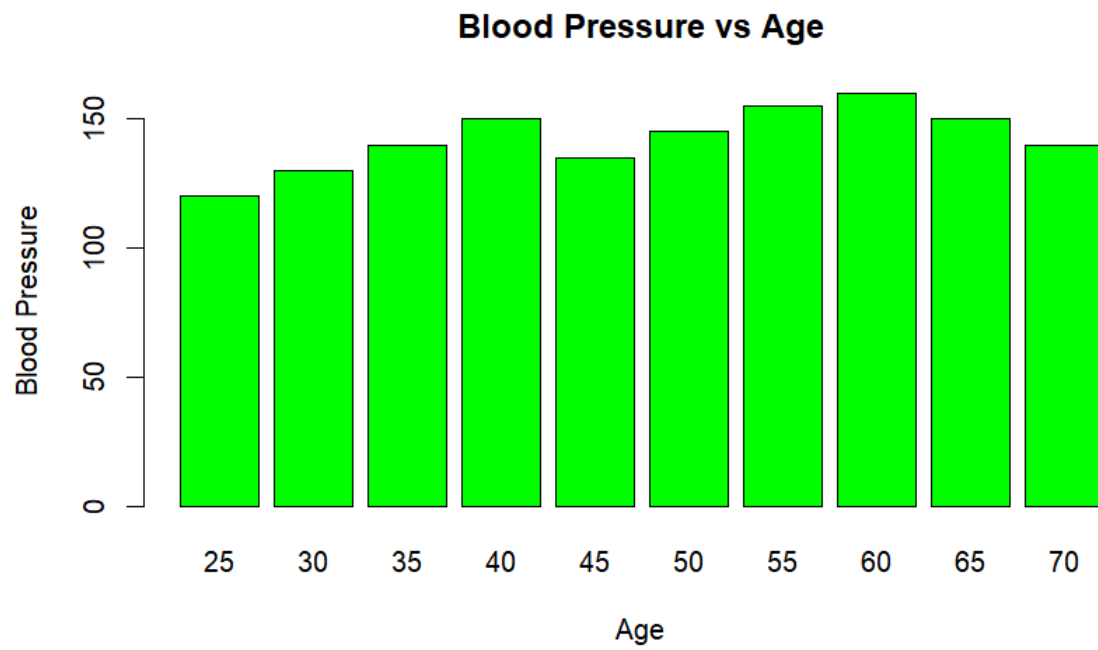
```
plot(data$Age, data$BloodPressure, main = "Blood Pressure vs Age", xlab = "Age", ylab = "Blood Pressure", pch = 16, col = "blue")
```

# Bar chart

```
barplot(data$BloodPressure, names.arg = data$Age, main = "Blood Pressure vs Age", xlab = "Age", ylab = "Blood Pressure", col = "green", border = "black")
```



**Output:**



DAY-03

04/03/24

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 true false false true

true true false false true

ouput:

## FP GROWTH

```
=== Run information ===

Scheme:      weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.5 -D 0.05 -U 3.0 -M 0.0
Relation:     database
Instances:    10
Attributes:   5
              a
              b
              c
              d
              e

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

FPGrowth found 8 rules (displaying top 8)

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)
4. [b=false]: 4 ==> [c=false]: 2   <conf:(0.5)> lift:(1) lev:(0) conv:(0.67)
5. [e=false]: 2 ==> [d=false]: 1   <conf:(0.5)> lift:(1.25) lev:(0.02) conv:(0.6)
6. [e=false]: 2 ==> [b=false]: 1   <conf:(0.5)> lift:(1.25) lev:(0.02) conv:(0.6)
7. [e=false]: 2 ==> [a=false]: 1   <conf:(0.5)> lift:(1.67) lev:(0.04) conv:(0.7)
8. [e=false]: 2 ==> [b=false, a=false]: 1   <conf:(0.5)> lift:(5) lev:(0.08) conv:(0.9)
```

## Apriori:

```
Apriori
=====

Minimum support: 0.45 (4 instances)
Minimum metric <confidence>: 0.5
Number of cycles performed: 11

Generated sets of large itemsets:

Size of set of large itemsets L(1): 8

Size of set of large itemsets L(2): 10

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

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 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 minsup > 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 dataset

@attribute milk{true,false}

@attribute beer{true,false}

@attribute diapers{true,false}

@attribute bread{true,false}

@attribute butter{true,false}

@attribute cookies{true,false}

@data

true true true false false false

true false false true true false

true false true false false true

false false false true true true

false true true false false true

true false true true true false

false false true true true false

false true true false false false

true false true true true false

false true false false false true

Ouput:

```
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=true 5 ==> beer=false 5    <conf:(1)> lift:(1.67) lev:(0.2) [2] conv:(2)
2. butter=true 5 ==> beer=false 5    <conf:(1)> lift:(1.67) lev:(0.2) [2] conv:(2)
3. butter=true 5 ==> bread=true 5    <conf:(1)> lift:(2) lev:(0.25) [2] conv:(2.5)
4. bread=true 5 ==> butter=true 5    <conf:(1)> lift:(2) lev:(0.25) [2] conv:(2.5)
5. butter=false 5 ==> bread=false 5   <conf:(1)> lift:(2) lev:(0.25) [2] conv:(2.5)
6. bread=false 5 ==> butter=false 5   <conf:(1)> lift:(2) lev:(0.25) [2] conv:(2.5)
7. bread=true butter=true 5 ==> beer=false 5    <conf:(1)> lift:(1.67) lev:(0.2) [2] conv:(2)
8. beer=false butter=true 5 ==> bread=true 5    <conf:(1)> lift:(2) lev:(0.25) [2] conv:(2.5)
9. beer=false bread=true 5 ==> butter=true 5    <conf:(1)> lift:(2) lev:(0.25) [2] conv:(2.5)
10. butter=true 5 ==> beer=false bread=true 5   <conf:(1)> lift:(2) lev:(0.25) [2] conv:(2.5)
```

Appriori:

Fp-growth:

```

=== Run information ===

Scheme:      weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.5 -D 0.05 -U 3.0 -M 0.0
Relation:    database
Instances:   10
Attributes:  6
             milk
             beer
             diapers
             bread
             butter
             cookies

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

FPGrowth found 10 rules (displaying top 10)

1. [milk=false, butter=false]: 3 ==> [bread=false]: 3   <conf:(1)> lift:(2) lev:(0.15) conv:(1.5)
2. [milk=false, bread=false]: 3 ==> [butter=false]: 3   <conf:(1)> lift:(2) lev:(0.15) conv:(1.5)
3. [milk=false]: 5 ==> [butter=false]: 3   <conf:(0.6)> lift:(1.2) lev:(0.05) conv:(0.83)
4. [butter=false]: 5 ==> [milk=false]: 3   <conf:(0.6)> lift:(1.2) lev:(0.05) conv:(0.83)
5. [milk=false]: 5 ==> [bread=false]: 3   <conf:(0.6)> lift:(1.2) lev:(0.05) conv:(0.83)
6. [bread=false]: 5 ==> [milk=false]: 3   <conf:(0.6)> lift:(1.2) lev:(0.05) conv:(0.83)
7. [milk=false]: 5 ==> [butter=false, bread=false]: 3   <conf:(0.6)> lift:(1.2) lev:(0.05) conv:(0.83)
8. [butter=false]: 5 ==> [milk=false, bread=false]: 3   <conf:(0.6)> lift:(2) lev:(0.15) conv:(1.17)
9. [bread=false]: 5 ==> [milk=false, butter=false]: 3   <conf:(0.6)> lift:(2) lev:(0.15) conv:(1.17)
10. [butter=false, bread=false]: 5 ==> [milk=false]: 3   <conf:(0.6)> lift:(1.2) lev:(0.05) conv:(0.83)

```

### 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 creit\_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

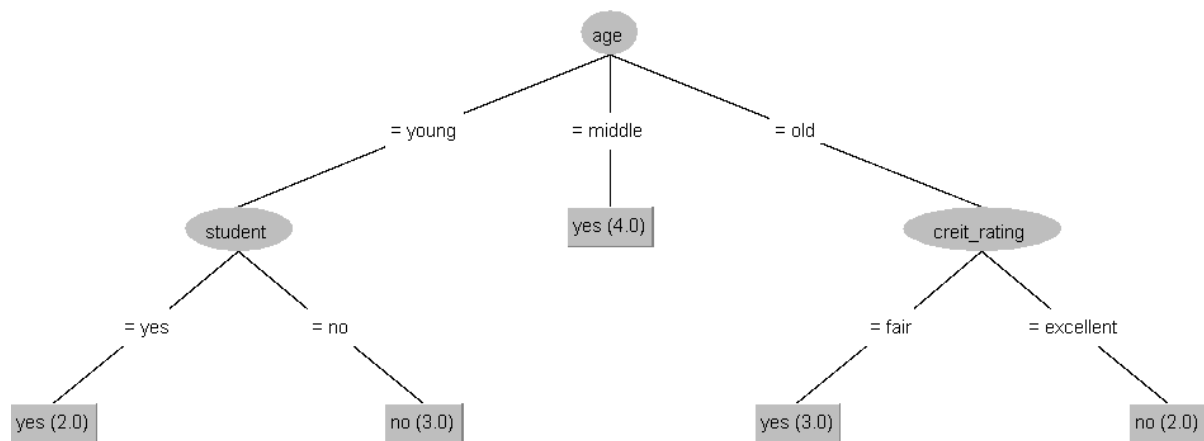
middle medium no excellent yes

middle high yes fair yes

old medium no excellent no

output:

decision tree:



Naïve bayes:

```

Correctly Classified Instances      7          50    %
Incorrectly Classified Instances    7          50    %
Kappa statistic                    -0.0426
Mean absolute error                 0.4167
Root mean squared error            0.5984
Relative absolute error             87.5    %
Root relative squared error        121.2987 %
Total Number of Instances         14

=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
      0.556   0.600   0.625    0.556   0.588     -0.043   0.633    0.758    yes
      0.400   0.444   0.333    0.400   0.364     -0.043   0.633    0.457    no
Weighted Avg.   0.500   0.544   0.521    0.500   0.508     -0.043   0.633    0.650

=== Confusion Matrix ===

 a b   <-- classified as
 5 4 | a = yes
 3 2 | b = no

```

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

Output:



```

Correctly Classified Instances      579           75.3906 %
Incorrectly Classified Instances    189           24.6094 %
Kappa statistic                     0.4484
Mean absolute error                 0.2955
Root mean squared error            0.4215
Relative absolute error             65.0135 %
Root relative squared error        88.4274 %
Total Number of Instances          768

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0.832   0.392   0.798     0.832   0.815     0.449   0.793   0.850   tested_negative
                0.608   0.168   0.660     0.608   0.633     0.449   0.793   0.667   tested_positive
Weighted Avg.   0.754   0.314   0.750     0.754   0.751     0.449   0.793   0.786

=== Confusion Matrix ===

  a    b  <-- classified as
416  84 |  a = tested_negative
105 163 |  b = tested_positive

```

5. 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

Input:

@relation dataset

@attribute M{true,false}

@attribute O{true,false}

@attribute N{true,false}

@attribute K{true,false}

@attribute E{true,false}

@attribute Y{true,false}

@attribute D{true,false}

@attribute A{true,false}

@attribute U{true,false}

@attribute C{true,false}

@attribute I{true,false}

@data

true true true true true true false false false false false

false true true true true true true false false false false

true false false true true false false false true false false

true false false true false true false false false true true false

false true false true true false false false false true true

output:

Apriori:

```
Apriori
=====

Minimum support: 0.85 (4 instances)
Minimum metric <confidence>: 0.6
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): 10
Size of set of large itemsets L(3): 6
Size of set of large itemsets L(4): 1

Best rules found:

1. A=false 5 ==> K=true 5    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
2. K=true 5 ==> A=false 5    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
3. E=true 4 ==> K=true 4    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
4. D=false 4 ==> K=true 4    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
5. U=false 4 ==> K=true 4    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
6. C=false 4 ==> K=true 4    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
7. E=true 4 ==> A=false 4    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
8. C=false 4 ==> E=true 4    <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
9. E=true 4 ==> C=false 4    <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
10. D=false 4 ==> A=false 4    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
```

Fpgrowth:

```

=== Run information ===

Scheme:      weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.6 -D 0.05 -U 1.0
Relation:    dataset
Instances:    5
Attributes:   11
              M
              O
              N
              K
              E
              Y
              D
              A
              U
              C
              I

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

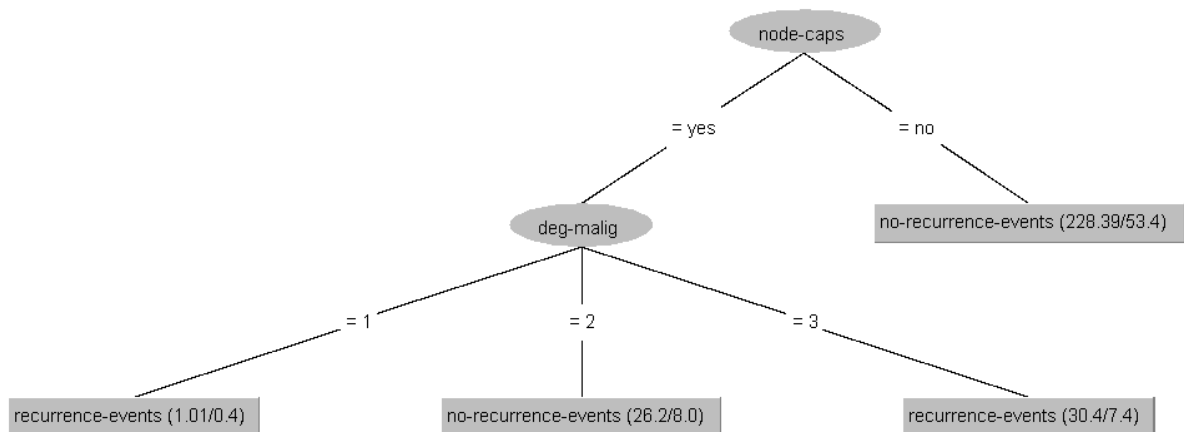
FPGrowth found 50 rules (displaying top 10)

1. [U=false]: 4 ==> [A=false]: 4 <conf:(1)> lift:(1) lev:(0) conv:(0)
2. [D=false]: 4 ==> [A=false]: 4 <conf:(1)> lift:(1) lev:(0) conv:(0)
3. [C=false]: 4 ==> [A=false]: 4 <conf:(1)> lift:(1) lev:(0) conv:(0)
4. [N=false]: 3 ==> [A=false]: 3 <conf:(1)> lift:(1) lev:(0) conv:(0)
5. [I=false]: 3 ==> [A=false]: 3 <conf:(1)> lift:(1) lev:(0) conv:(0)
6. [N=false]: 3 ==> [D=false]: 3 <conf:(1)> lift:(1.25) lev:(0.12) conv:(0.6)
7. [I=false]: 3 ==> [C=false]: 3 <conf:(1)> lift:(1.25) lev:(0.12) conv:(0.6)
8. [U=false, D=false]: 3 ==> [A=false]: 3 <conf:(1)> lift:(1) lev:(0) conv:(0)
9. [U=false, C=false]: 3 ==> [A=false]: 3 <conf:(1)> lift:(1) lev:(0) conv:(0)
10. [D=false, C=false]: 3 ==> [A=false]: 3 <conf:(1)> lift:(1) lev:(0) conv:(0)

```

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

Output:



Logistic:

```

Correctly Classified Instances      217                75.8741 %
Incorrectly Classified Instances    69                24.1259 %
Kappa statistic                    0.2899
Mean absolute error                0.3658
Root mean squared error           0.4269
Relative absolute error            87.4491 %
Root relative squared error       93.4017 %
Total Number of Instances         286

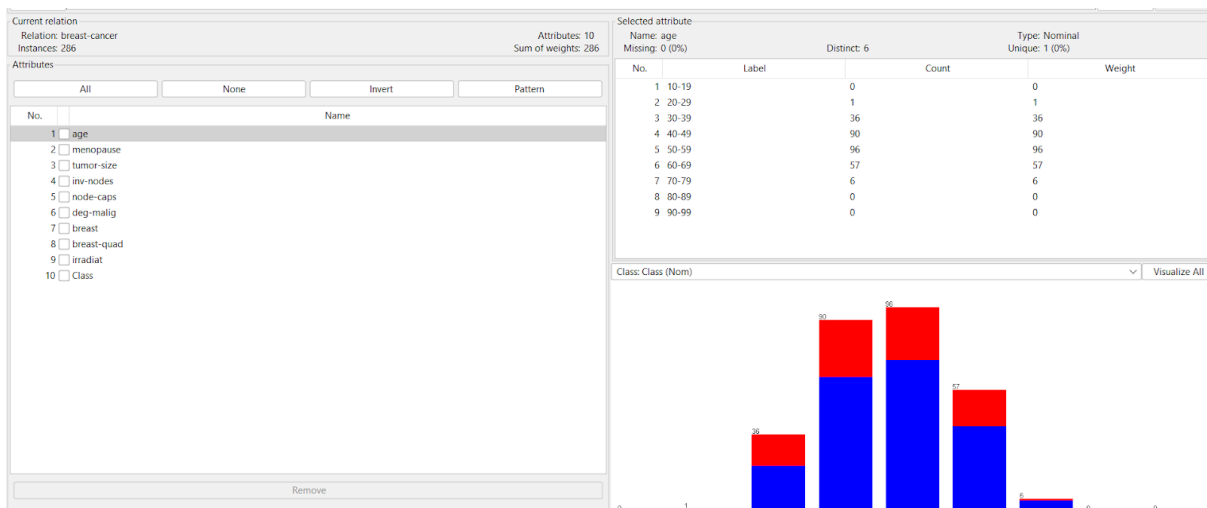
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC       ROC Area  PRC Area  Class
                0.965   0.729   0.758     0.965   0.849     0.352     0.639    0.767    no-recurrence-events
                0.271   0.035   0.767     0.271   0.400     0.352     0.639    0.461    recurrence-events
Weighted Avg.   0.759   0.523   0.760     0.759   0.716     0.352     0.639    0.676

=== Confusion Matrix ===

  a    b  <-- classified as
194    7 |  a = no-recurrence-events
 62   23 |  b = recurrence-events

```



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 dataset

@attribute hotdogs{true,false}

@attribute buns{true,false}

@attribute ketchup{true,false}

@attribute coke{true,false}

@attribute chips{true,false}

@data

true true true false false

true true false false false

true false false true true

false false false true true

false false true false true

true false false true true

output:

Fp growth:

```
=== Run information ===

Scheme:      weka.associations.FPFGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1
Relation:    dataset
Instances:   6
Attributes:  5
             hotdogs
             buns
             ketchup
             coke
             chips

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

FPFGrowth found 7 rules (displaying top 7)

1. [hotdogs=false]: 2 ==> [buns=false]: 2   <conf:(1)> lift:(1.5) lev:(0.11) conv:(0.67)
2. [chips=false]: 2 ==> [coke=false]: 2     <conf:(1)> lift:(2) lev:(0.17) conv:(1)
3. [ketchup=false, hotdogs=false]: 1 ==> [buns=false]: 1   <conf:(1)> lift:(1.5) lev:(0.06) conv:(0.33)
4. [ketchup=false, coke=false]: 1 ==> [chips=false]: 1     <conf:(1)> lift:(3) lev:(0.11) conv:(0.67)
5. [ketchup=false, chips=false]: 1 ==> [coke=false]: 1     <conf:(1)> lift:(2) lev:(0.08) conv:(0.5)
6. [buns=false, coke=false]: 1 ==> [hotdogs=false]: 1     <conf:(1)> lift:(3) lev:(0.11) conv:(0.67)
7. [coke=false, hotdogs=false]: 1 ==> [buns=false]: 1     <conf:(1)> lift:(1.5) lev:(0.06) conv:(0.33)
```

Apriori:

```

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

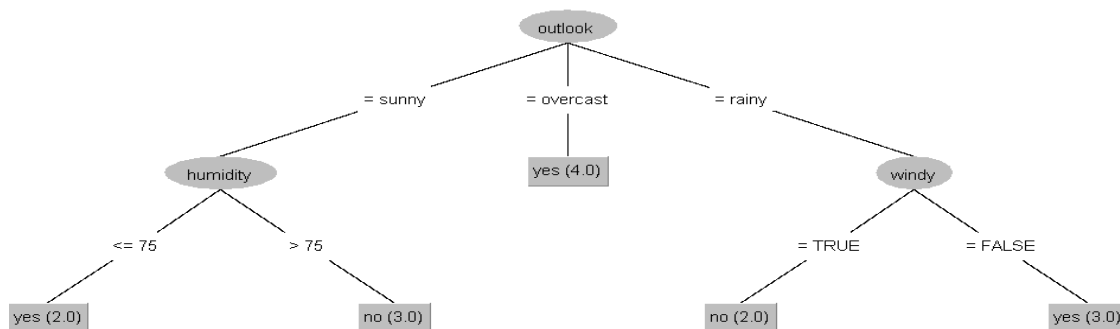
Size of set of large itemsets L(4): 1

Best rules found:

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

```

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



#### Decision Table:

Number of training instances: 14

Number of Rules : 1

Non matches covered by Majority class.

Best first.

Start set: no attributes

Search direction: forward

Stale search after 5 node expansions

Total number of subsets evaluated: 13

Merit of best subset found: 64.286

Evaluation (for feature selection): CV (leave one out)

Feature set: 5

Time taken to build model: 0.01 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances	9	64.2857 %
Incorrectly Classified Instances	5	35.7143 %
Kappa statistic	0	
Mean absolute error	0.4524	
Root mean squared error	0.4797	
Relative absolute error	97.4359 %	
Root relative squared error	100.0539 %	
Total Number of Instances	14	

#### DAY4 05/03/2024

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

### Input:

@relation dataset

@attribute customerid{1,2,3,4,5}

@attribute gender{male,female}

@attribute age{19,21,20,23,31}

@attribute income{15,16,17}

@attribute score{usual,priority,senior,young}

@data

1 male 19 15 usual

2 male 21 15 priority

3 female 20 16 young

4 female 23 16 priority

5 female 31 17 senior

### Output:



```

kMeans
=====

Number of iterations: 2
Within cluster sum of squared errors: 10.0

Initial starting points (random):

Cluster 0: 4,female,23,16,priority
Cluster 1: 2,male,21,15,priority

Missing values globally replaced with mean/mode

Final cluster centroids:

Attribute      Full Data      Cluster#
                (5.0)      (3.0)      (2.0)
=====
customerid      1          3          1
gender          female    female    male
age             19         20         19
income          15         16         15
score           priority  priority  usual

Time taken to build model (full training data) : 0.01 seconds

=== Evaluation on test set ===
Clustered Instances

0          3 ( 60%)
1          2 ( 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 the cluster.

EmployeeID	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

### Input:

@relation dataset

@attribute id{1,2,3,4,5,6}

@attribute gender{male,female}

@attribute age{28,25,26,30,29}

@attribute salary{15,16,17,20}

@attribute credit{39,27,42,40,64,72}

@data

1 male 28 15 39

2 male 25 15 27

3 female 26 16 42

4 female 25 16 40

5 female 30 17 64

6 male 29 20 72

### Output:

```
kMeans
=====

Number of iterations: 2
Within cluster sum of squared errors: 14.0

Initial starting points (random):

Cluster 0: 4,female,25,16,40
Cluster 1: 1,male,28,15,39

Missing values globally replaced with mean/mode

Final cluster centroids:

Attribute      Full Data      Cluster#
                (6.0)      (3.0)      (3.0)
=====
id              1              3              1
gender          male        female        male
age             25          25          28
salary          15          16          15
credit          39          42          39

Time taken to build model (full training data) : 0 seconds

=== Evaluation on test set ===
Clustered Instances

0          3 ( 50%)
1          3 ( 50%)
```

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.

# Naive Bayes Classifier

Attribute	Class	
	yes	no
	(0.63)	(0.38)
=====		
outlook		
sunny	3.0	4.0
overcast	5.0	1.0
rainy	4.0	3.0
[total]	12.0	8.0
temperature		
mean	72.9697	74.8364
std. dev.	5.2304	7.384
weight sum	9	5
precision	1.9091	1.9091
humidity		
mean	78.8395	86.1111
std. dev.	9.8023	9.2424
weight sum	9	5
precision	3.4444	3.4444
windy		
TRUE	4.0	4.0
FALSE	7.0	3.0
[total]	11.0	7.0

```

Time taken to build model: 0.01 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances      13           92.8571 %
Incorrectly Classified Instances    1            7.1429 %
Kappa statistic                     0.8372
Mean absolute error                 0.2798
Root mean squared error            0.3315
Relative absolute error             60.2576 %
Root relative squared error        69.1352 %
Total Number of Instances          14

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
          1.000    0.200    0.900    1.000    0.947      0.849    0.911    0.947    yes
          0.800    0.000    1.000    0.800    0.889      0.849    0.911    0.911    no
Weighted Avg.    0.929    0.129    0.936    0.929    0.926      0.849    0.911    0.934

=== Confusion Matrix ===

a b   <-- classified as
9 0 | a = yes
1 4 | b = no

```

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

**Input:**

@relation dataset

@attribute person {gopu,babu,baby,gopal,krishna,jai,dev,malini,hema,anu}

@attribute vegetarian {yes,no}

@data

gopu yes

babu yes

baby yes

gopal no

krishna yes

jai no

dev no

malini yes

hema yes

anu yes

### Output:

```
Time taken to build model: 0 seconds

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

Correctly Classified Instances      7           70    %
Incorrectly Classified Instances    3           30    %
Kappa statistic                     0
Mean absolute error                 0.48
Root mean squared error             0.502
Relative absolute error             101.5385 %
Root relative squared error         100.1512 %
Total Number of Instances          10

=== Detailed Accuracy By Class ===

              TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
              1.000    1.000    0.700     1.000    0.824      ?       0.071    0.636    yes
              0.000    0.000    ?         0.000    ?         ?       0.071    0.300    no
Weighted Avg.   0.700    0.700    ?         0.700    ?         ?       0.071    0.535

=== Confusion Matrix ===

a b   <-- classified as
7 0 | a = yes
3 0 | b = no
```

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

**Input:**

@relation dataset

@attribute x {4,1,5,7,10,2,50,25,90,36}

@attribute y {12,5,13,19,31,7,153,72,275,110}

@data

4,12

1,5

5,13

7,19

10,31

2,7

50,153

25,72

90,275

36,110

R programming:

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

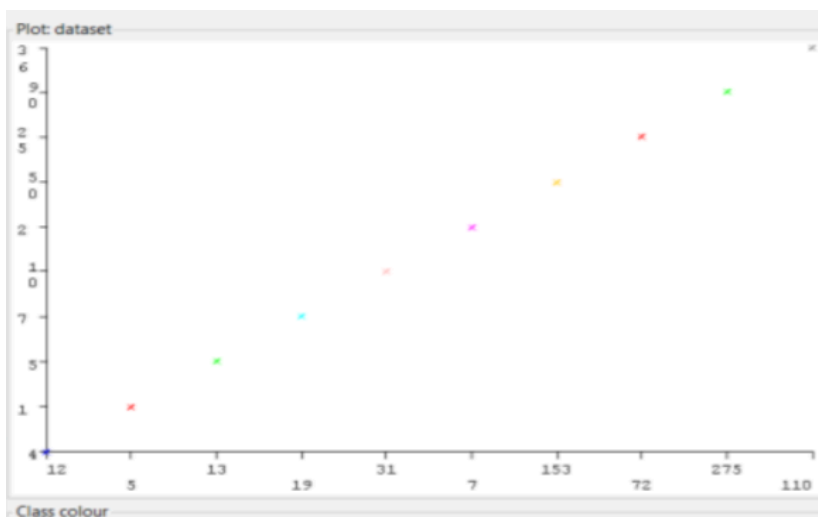
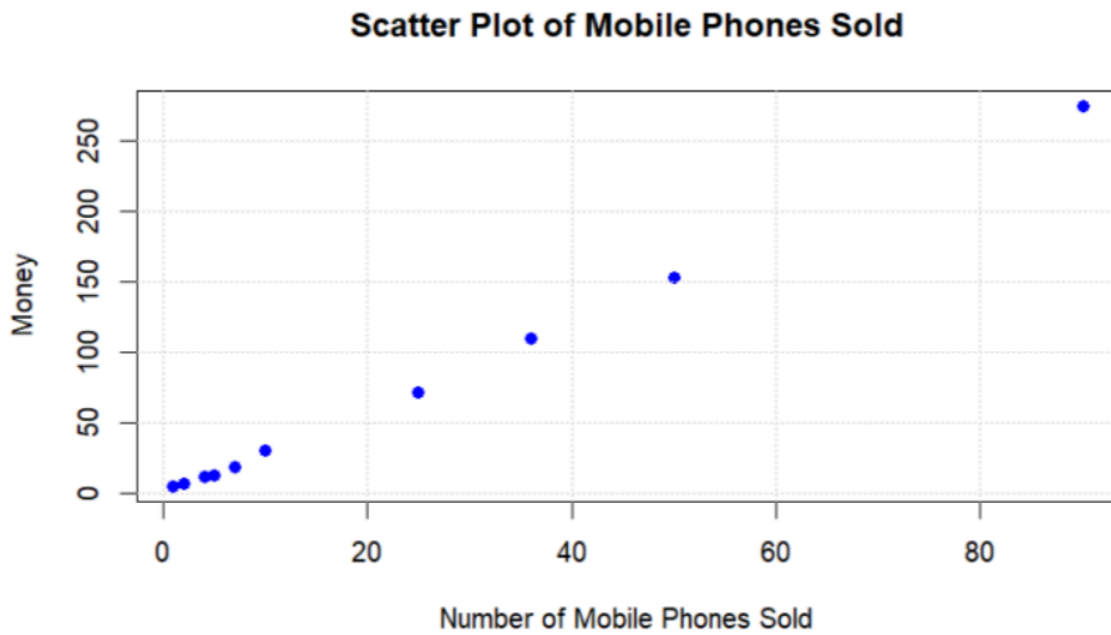
```
y_values <- c(12, 5, 13, 19, 31, 7, 153, 72, 275, 110)plot(x_values, y_values, main="Scatter  
Plot of Mobile Phones Sold",
```

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

```
pch=16, col="blue")
```

```
grid()
```

Output:



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

**Input:**

@relation dataset

@attribute transid{1,2,3,4,5}

@attribute bread{true,false}

@attribute cheese{true,false}

@attribute egg{true,false}

@attribute juice{true,false}

@attribute milk{true,false}

@attribute yogurt{true,false}

@data

1,true,true,true,true,false,false

2,true,true,false,true,false,false

3,true,false,false,false,true,true

4,true,false,false,true,true,false

5,false,true,false,true,true,falseoutput”

Output:



**7.**Prediction of Diabetes Data using Decision tree classifier in WEKA. Compare it with Support Vector Machine classifier. Show the result accuracy and F1 measure calculation .Plot the graph and explain the summary of results.

8.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

(c) clustering

**Input:**

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

```
bins_a <- cut(marks, breaks = 3, labels = c("Low", "Medium", "High"))
```

```
bins_b <- cut(marks, breaks = seq(min(marks), max(marks), length.out = 4), labels =  
c("Low", "Medium", "High"))
```

```
k <- 3
```

```
clusters <- kmeans(matrix(marks), centers = k)
```

```
bins_c <- cut(clusters$centers[clusters$cluster], breaks = 3, labels = c("Low", "Medium",  
"High"))
```

```
par(mfrow = c(1, 3))
```

```
hist(marks, main = "Equal-frequency (equi-depth) partitioning", col = "skyblue", breaks = 3)
```

```
hist(marks, main = "Equal-width partitioning", col = "lightgreen", breaks = seq(min(marks),  
max(marks), length.out = 4))
```

```
hist(marks, main = "Clustering", col = "lightpink", breaks = 3)
```

Output:

```

weight sum          500          268
precision           0.0045       0.0045

age
mean                31.2494      37.0808
std. dev.           11.6059      10.9146
weight sum          500          268
precision           1.1765       1.1765

Time taken to build model: 0.01 seconds

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

Correctly Classified Instances      586          76.2021 %
Incorrectly Classified Instances    182          23.6979 %
Kappa statistic                    0.4664
Mean absolute error                 0.2841
Root mean squared error             0.4168
Relative absolute error             62.5028 %
Root relative squared error         87.4349 %
Total Number of Instances          768

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MDC      ROC Area  FRC Area  Class
0.844   0.388   0.802   0.844   0.823   0.468   0.819   0.892   tested_negative
0.612   0.156   0.678   0.612   0.643   0.468   0.819   0.671   tested_positive
Weighted Avg.   0.763   0.307   0.755   0.763   0.760   0.468   0.819   0.815

=== Confusion Matrix ===

  a  b  <-- classified as
422  78 |  a = tested_negative
104 164 |  b = tested_positive

```

```

Number of Leaves :      20

Size of the tree :      39

Time taken to build model: 0.05 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  MDC      ROC Area  FRC 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

```

8. 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

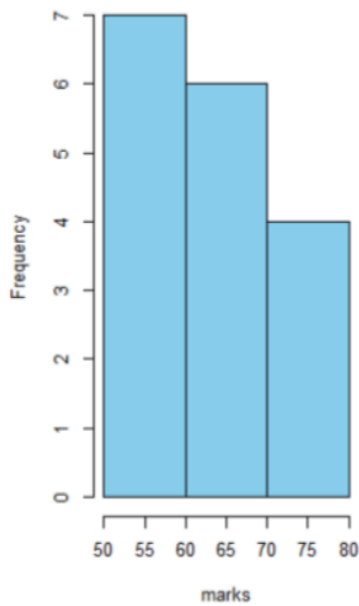
(c) clustering

**Input:**

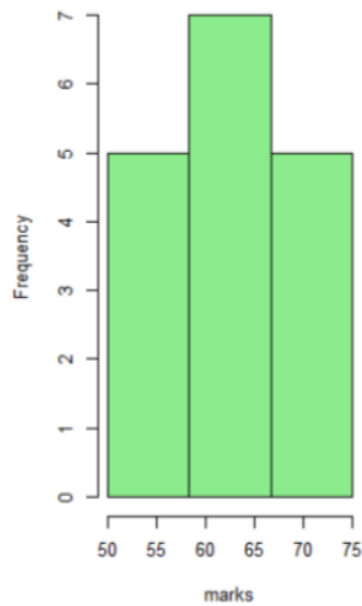
```
marks <- c(55, 60, 71, 63, 55, 65, 50, 55, 58, 59, 61, 63, 65,  
bins_a <- cut(marks, breaks = 3, labels = c("Low", "Medium", "High"))  
bins_b <- cut(marks, breaks = seq(min(marks), max(marks), length.out = 4), labels =  
c("Low", "Medium", "High"))  
k <- 3  
clusters <- kmeans(matrix(marks), centers = k)  
bins_c <- cut(clusters$centers[clusters$cluster], breaks = 3, labels = c("Low", "Medium",  
"High"))  
par(mfrow = c(1, 3))  
hist(marks, main = "Equal-frequency (equi-depth) partitioning", col = "skyblue", breaks = 3)  
hist(marks, main = "Equal-width partitioning", col = "lightgreen", breaks = seq(min(marks),  
max(marks), length.out = 4))  
hist(marks, main = "Clustering", col = "lightpink", breaks = 3)
```

**Output:**

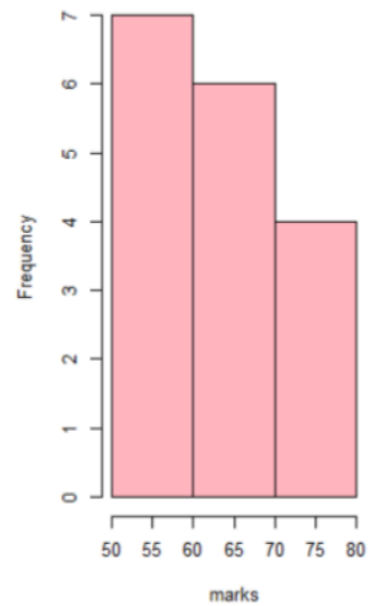
Equal-frequency (equi-depth) partition



Equal-width partitioning

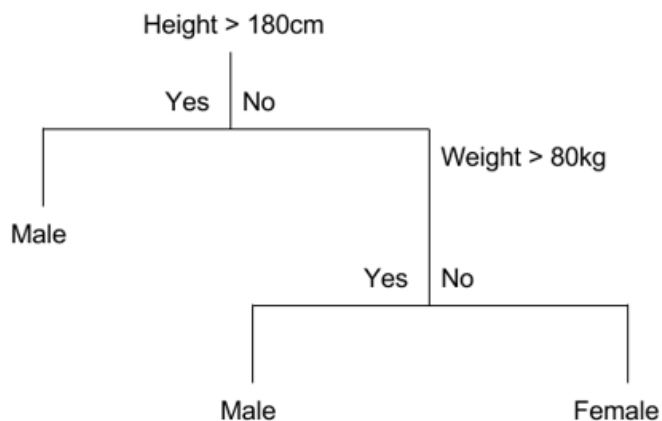


Clustering



9. Consider this Decision tree :

- create the data set for the below tree using ARFF format and calculate accuracy and decision for the same
- Using this decision tree generate the rules based on rule based induction.
- Compare both the algorithms and plot the confusion matrix.



Output:

10. Create an ARFF file for the table below and implement for the Apriori Algorithm and FP growth algorithm and compare the rules generated by both the algorithms. Identify the unique rules generated by the above algorithms.

NOTE: Assume Min\_sup=2 and confidence= 50%

T.ID	ITEMS
T1	SONY, BPL, LG
T2	BPL, SAMSUNG
T3	BPL, ONIDA
T4	SONY, BPL, SAMSUNG
T5	SONY, ONIDA
T6	BPL, ONIDA
T7	SONY, ONIDA
T8	SONY, BPL, ONIDA, LG
T9	SONY, BPL, ONIDA

**Input:**

@relation dataset

@attribute id {1,2,3,4,5,6,7,8,9}

@attribute sony {true,false}

@attribute bpl {true,false}

@attribute lg {true,false}

@attribute samsung {true,false}

@attribute onida {true,false}

@data

1,true,true,true,false,false

2,false,true,false,true,false

3,false,true,false,false,true

4,true,true,false,true,false

5,true,false,false,false,true

6,false,true,false,false,true

7,true,false,false,false,true

8,true,true,true,false,true

9,true,true,false,false,true

Output:

```
=== Run information ===

Scheme:      weka.associations.Apriori -N 10 -T 0 -C 50.0 -D 0.05 -U 2.0 -M 0.1 -S -1.0 -c -1
Relation:     dataset
Instances:    9
Attributes:   6
              id
              sony
              bpl
              lg
              samsung
              onida

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

Apriori
=====

Minimum support: 0.11 (1 instances)
Minimum metric <confidence>: 50
Number of cycles performed: 18

Generated sets of large itemsets:

Size of set of large itemsets L(1): 19
Size of set of large itemsets L(2): 78
Size of set of large itemsets L(3): 137
Size of set of large itemsets L(4): 120
Size of set of large itemsets L(5): 52
Size of set of large itemsets L(6): 9

Best rules found:
```

**11.** The given are the strike-rates scored by a batsman in season 1 in different tournaments.  
100, 70, 60, 90, 90

- (a) min-max normalization by setting min = 0 and max = 1
- (b) z-score normalization
- (c) z-score normalization using the mean absolute deviation instead of standard deviation
- (d) normalization by decimal scaling

**Input:**

```
strike_rates <- c(100, 70, 60, 90, 90)
```

```
min_max_normalization <- function(x) {
```

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

```

}

normalized_min_max <- min_max_normalization(strike_rates)

z_score_normalization <- function(x) {

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

}

normalized_z_score <- z_score_normalization(strike_rates)

mad_normalization <- function(x) {

(x - mean(x)) / mad(x)

}

normalized_mad <- mad_normalization(strike_rates)

decimal_scaling_normalization <- function(x) {

x / 10^(ceiling(log10(max(x))))

}

normalized_decimal_scaling <- decimal_scaling_normalization(strike_rates)

cat("Original Data:", strike_rates, "\n\n")

cat("(a) Min-Max Normalization:", normalized_min_max, "\n")

cat("(b) Z-Score Normalization:", normalized_z_score, "\n")

cat("(c) Z-Score Normalization (MAD):", normalized_mad, "\n")

cat("(d) Normalization by Decimal Scaling:", normalized_decimal_scaling, "\n")

```

Output:

```

Console Terminal x Background Jobs x
R 4.3.2 · ~/
> strike_rates <- c(100, 70, 60, 90, 90)
> min_max_normalization <- function(x) {
+   (x - min(x)) / (max(x) - min(x))
+ }
> normalized_min_max <- min_max_normalization(strike_rates)
> z_score_normalization <- function(x) {
+   (x - mean(x)) / sd(x)
+ }
> normalized_z_score <- z_score_normalization(strike_rates)
> mad_normalization <- function(x) {
+   (x - mean(x)) / mad(x)
+ }
> normalized_mad <- mad_normalization(strike_rates)
> decimal_scaling_normalization <- function(x) {
+   x / 10^(ceiling(log10(max(x))))
+ }
> normalized_decimal_scaling <- decimal_scaling_normalization(strike_rates)
> cat("Original Data:", strike_rates, "\n\n")
Original Data: 100 70 60 90 90

> cat("(a) Min-Max Normalization:", normalized_min_max, "\n")
(a) Min-Max Normalization: 1 0.25 0 0.75 0.75
> cat("(b) Z-Score Normalization:", normalized_z_score, "\n")
(b) Z-Score Normalization: 1.095445 -0.7302967 -1.338877 0.4868645 0.4868645
> cat("(c) Z-Score Normalization (MAD):", normalized_mad, "\n")
(c) Z-Score Normalization (MAD): 1.214083 -0.8093889 -1.48388 0.5395926 0.5395926
> cat("(d) Normalization by Decimal Scaling:", normalized_decimal_scaling, "\n")
(d) Normalization by Decimal Scaling: 1 0.7 0.6 0.9 0.9
> |

```

12. Suppose some car is tested for the AvgSpeed and TotalTime data for 9 randomly selected car with the following result

AvgSpeed (in kph)	78	81	82	74	83	82	77	80	70
TotalTime (in mins)	39	37	36	42	35	36	40	38	46

- Calculate the standard deviation of AvgSpeed and TotalTime.
- Calculate the Variance of AvgSpeed and TotalTime for the above dataset.

**Input:**

```
avg_speed <- c(78, 81, 82, 74, 83, 82, 77, 80, 70)
```



```

total_time <- c(39, 37, 36, 42, 35, 36, 40, 38, 46)

sd_avg_speed <- sd(avg_speed)

sd_total_time <- sd(total_time)

var_avg_speed <- var(avg_speed)

var_total_time <- var(total_time)

cat("Standard Deviation of AvgSpeed:", sd_avg_speed, "\n")

cat("Standard Deviation of TotalTime:", sd_total_time, "\n\n")

cat("Variance of AvgSpeed:", var_avg_speed, "\n")

cat("Variance of TotalTime:", var_total_time, "\n")

```

Output:

13. Consider the table

**c) TID    items bought**

**d) T100   {M, O, N, K, E, Y}**

**e) T200   {D, O, N, K, E, Y }**

**f) T300   {M, A, K, E}**

**g) T400   {M, U, C, K, Y}**

**h) T500   {C, O, O, K, I ,E}**

- i) (a) Find all frequent item set using Apriori and FP-growth, respectively. Compare the efficiency of the two mining processes.
- j) (b) List all of the strong association rules (with support  $s$  and confidence  $c$ ) matching the following metarule, where  $X$  is a variable representing customers, and  $item_i$  denotes variables representing items (e.g., "A", "B", etc.):
- k)  $\forall x \in \text{transaction}, \text{buys}(X, \text{item1}) \wedge \text{buys}(X, \text{item2}) \Rightarrow \text{buys}(X, \text{item3})$

Input:

@relation dataset

@attribute M{true,false}

@attribute O{true,false}

@attribute N{true,false}

@attribute K{true,false}

@attribute E{true,false}

@attribute Y{true,false}

@attribute D{true,false}

@attribute A{true,false}

@attribute U{true,false}

@attribute C{true,false}

@attribute I{true,false}

@data

true true true true true true false false false false false false

false true true true true true true false false false false false

true false false true true false false false true false false false

true false false true false true false false false true true false

false true false true true false false false false false true true

Output:

Preprocess   Classify   Cluster   **Associate**   Select attributes   Visualize

---

Associator

Choose **FPGrowth** -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1

Start   Stop

Result list (right-click for ...)

- 09:07:14 - Apriori
- 09:07:27 - 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 0.1
Relation:     dataset
Instances:    5
Attributes:   11
              M
              O
              N
              K
              E
              Y
              D
              A
              U
              C
              I

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

FPGrowth found 16 rules (displaying top 10)

1. [U=false]: 4 ==> [A=false]: 4   <conf:(1)> lift:(1) lev:(0) conv:(0)
2. [D=false]: 4 ==> [A=false]: 4   <conf:(1)> lift:(1) lev:(0) conv:(0)
3. [C=false]: 4 ==> [A=false]: 4   <conf:(1)> lift:(1) lev:(0) conv:(0)
4. [N=false]: 3 ==> [A=false]: 3   <conf:(1)> lift:(1) lev:(0) conv:(0)
5. [I=false]: 3 ==> [A=false]: 3   <conf:(1)> lift:(1) lev:(0) conv:(0)
6. [N=false]: 3 ==> [D=false]: 3   <conf:(1)> lift:(1.25) lev:(0.12) conv:(0.6)
7. [I=false]: 3 ==> [C=false]: 3   <conf:(1)> lift:(1.25) lev:(0.12) conv:(0.6)
8. [U=false, D=false]: 3 ==> [A=false]: 3   <conf:(1)> lift:(1) lev:(0) conv:(0)
9. [U=false, C=false]: 3 ==> [A=false]: 3   <conf:(1)> lift:(1) lev:(0) conv:(0)
10. [D=false, C=false]: 3 ==> [A=false]: 3   <conf:(1)> lift:(1) lev:(0) conv:(0)

```

Associator

Choose **FPGrowth** -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1

Start   Stop

Result list (right-click for ...)

- 09:07:14 - Apriori
- 09:07:27 - FPGrowth

Associator output

```

=== 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): 10

Size of set of large itemsets L(3): 6

Size of set of large itemsets L(4): 1

Best rules found:

1. A=false 5 ==> K=true 5   <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
2. K=true 5 ==> A=false 5   <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
3. E=true 4 ==> K=true 4   <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
4. D=false 4 ==> K=true 4   <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
5. U=false 4 ==> K=true 4   <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
6. C=false 4 ==> K=true 4   <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
7. E=true 4 ==> A=false 4   <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
8. C=false 4 ==> E=true 4   <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
9. E=true 4 ==> C=false 4   <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
10. D=false 4 ==> A=false 4   <conf:(1)> lift:(1) lev:(0) [0] conv:(0)

```

