**Ex.No**.5 **ASSOCIATION RULE MINING – APRIORI ALGORITHM**

**Date**: 29-08-23

**Aim**

To implement Apriori algorithm in finding the frequent data sets through R programming.

**Procedure**

1. To do programming in R, first install “RStudio” and “R” in the system. RStudio is an integrated development environment [IDE] for R and python.
2. Select the File in taskbar →open New file →R script or use shortcut “ctrl+shift+N”
3. Install the ‘arules,arulesViz, RColorBrewer’ package and load it in R.
4. Import the built-in dataset ‘groceries’.
5. Apply the Apriori Algorithm on the groceries dataset.
6. Write the program in the script and save it using the extension R.
7. Run the program by clicking Run option or use the shortcut “ctrl+enter”.
8. See the output in the console tab.

**Concepts Involved**

* Applying the Apriori’s algorithm -Association rule mining on the data set.

**APRIORI ALGORITHM**

Apriori algorithm is related to the frequent itemset generation. The primary requirements to find the association rules in data mining are:

* **Brute Force:** Analyze all the rules and find the support and confidence levels for the individual rule. Afterward, eliminate the values which are less than the threshold support and confidence levels.

1. **Generate:**Create a table containing support count of each item present in dataset – Called candidate set.(C1)

Compare candidate set item’s support count with minimum support count. This gives us item set.(L1)

1. **Join :**

Generate candidate set C2 using L1 (this is called join step). Condition of joining Lk-1 and Lk-1 is that it should have (K-2) elements in common.

1. **Prune:**

Check all subsets of an itemset are frequent or not and if not frequent remove that itemset.

compare candidate (C2) support count with minimum support count(here min\_support=2 if support\_count of candidate set item is less than min\_support then remove those items) this gives us itemset L2.

Like this, do the same process until there are no frequent data sets.

**Script**

**#create a list of basket**

market\_basket <-

list(

c("apple", "beer", "rice", "meat"),

c("apple", "beer", "rice"),

c("apple", "beer"),

c("apple", "pear"),

c("milk", "beer", "rice", "meat"),

c("milk", "beer", "rice"),

c("milk", "beer"),

c("milk", "pear"))

#set transaction names

names(market\_basket) <- paste("T", c(1:8), sep = "")

#transform data

trans <- as(market\_basket, "transactions")

#inspect data

dim(trans)

itemLabels(trans)

summary(trans)

image(trans)

**Output**

[1] 8 6

[1] "apple" "beer" "meat" "milk" "pear" "rice"

transactions as itemMatrix in sparse format with

8 rows (elements/itemsets/transactions) and

6 columns (items) and a density of 0.4583333

most frequent items:

beer apple milk rice meat (Other)

6 4 4 4 2 2

element (itemset/transaction) length distribution:

sizes

2 3 4

4 2 2

Min. 1st Qu. Median Mean 3rd Qu. Max.

2.00 2.00 2.50 2.75 3.25 4.00

includes extended item information - examples:

labels

1 apple

2 beer

3 meat

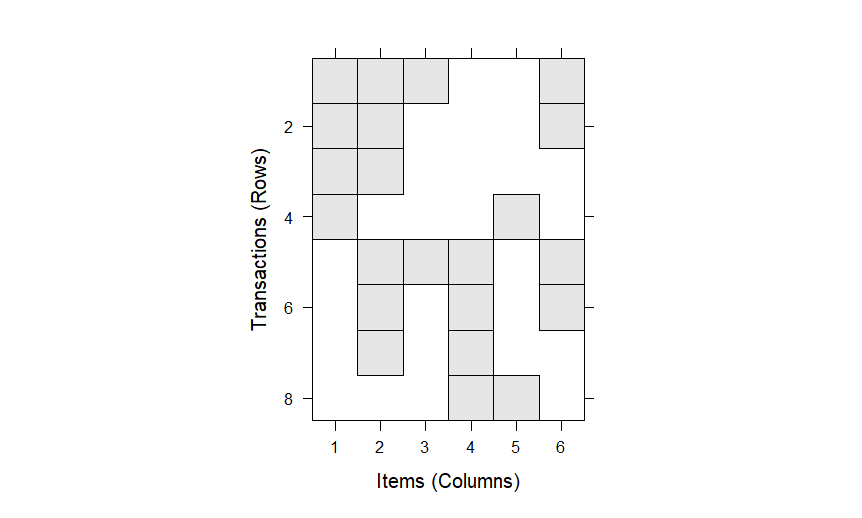
includes extended transaction information - examples:

transactionID

1 T1

2 T2

3 T3

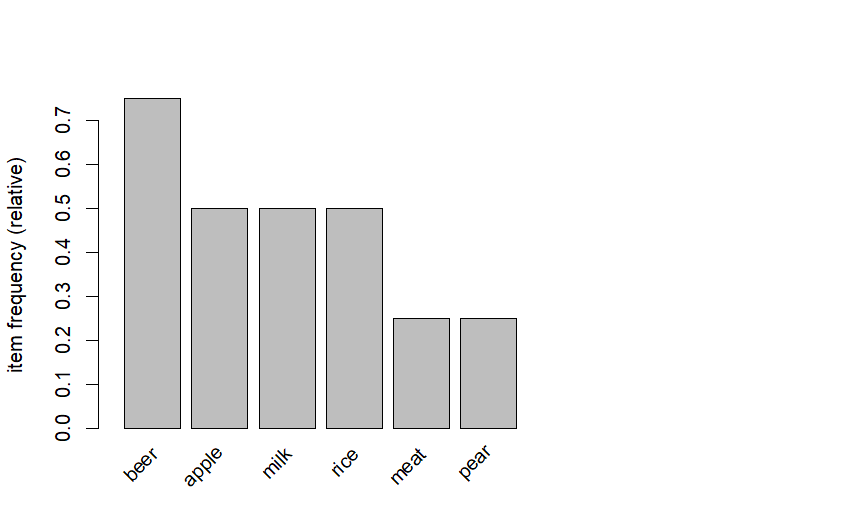
****

**Script**

#display relative item frequency

itemFrequencyPlot(trans, topN=10, cex.names=1)

**Output**

****

**Script**

#min support 0.3, min confidence 0.5

rules <- apriori(trans,

parameter = list(supp=0.3, conf=0.5,

maxlen=10,

target= "rules"))

summary(rules)

inspect(rules)

**Output**

> summary(rules)

set of 10 rules

rule length distribution (lhs + rhs):sizes

1 2

4 6

Min. 1st Qu. Median Mean 3rd Qu. Max.

1.0 1.0 2.0 1.6 2.0 2.0

summary of quality measures:

support confidence coverage lift count

Min. :0.375 Min. :0.5000 Min. :0.5000 Min. :1.000 Min. :3.0

1st Qu.:0.375 1st Qu.:0.5000 1st Qu.:0.5625 1st Qu.:1.000 1st Qu.:3.0

Median :0.500 Median :0.5833 Median :0.7500 Median :1.000 Median :4.0

Mean :0.475 Mean :0.6417 Mean :0.7750 Mean :1.067 Mean :3.8

3rd Qu.:0.500 3rd Qu.:0.7500 3rd Qu.:1.0000 3rd Qu.:1.000 3rd Qu.:4.0

Max. :0.750 Max. :1.0000 Max. :1.0000 Max. :1.333 Max. :6.0

mining info:

data ntransactions support confidence

trans 8 0.3 0.5

call

apriori(data = trans, parameter = list(supp = 0.3, conf = 0.5, maxlen = 10, target = "rules"))

> inspect(rules)

lhs rhs support confidence coverage lift count

[1] {} => {apple} 0.500 0.5000000 1.00 1.000000 4

[2] {} => {milk} 0.500 0.5000000 1.00 1.000000 4

[3] {} => {rice} 0.500 0.5000000 1.00 1.000000 4

[4] {} => {beer} 0.750 0.7500000 1.00 1.000000 6

[5] {apple} => {beer} 0.375 0.7500000 0.50 1.000000 3

[6] {beer} => {apple} 0.375 0.5000000 0.75 1.000000 3

[7] {milk} => {beer} 0.375 0.7500000 0.50 1.000000 3

[8] {beer} => {milk} 0.375 0.5000000 0.75 1.000000 3

[9] {rice} => {beer} 0.500 1.0000000 0.50 1.333333 4

[10] {beer} => {rice} 0.500 0.6666667 0.75 1.333333 4

**Script**

#Min Support 0.3, confidence as 0.5.

rules <- apriori(trans,

parameter = list(supp=0.3, conf=0.5,

maxlen=10,

minlen=2,

target= "rules"))

inspect(rules)

**Output**

> inspect(rules)

lhs rhs support confidence coverage lift count

[1] {apple} => {beer} 0.375 0.7500000 0.50 1.000000 3

[2] {beer} => {apple} 0.375 0.5000000 0.75 1.000000 3

[3] {milk} => {beer} 0.375 0.7500000 0.50 1.000000 3

[4] {beer} => {milk} 0.375 0.5000000 0.75 1.000000 3

[5] {rice} => {beer} 0.500 1.0000000 0.50 1.333333 4

[6] {beer} => {rice} 0.500 0.6666667 0.75 1.333333 4

**Script**

#Set LHS and RHS

beer\_rules\_rhs <- apriori(trans,

parameter = list(supp=0.3, conf=0.5,

maxlen=10,

minlen=2),

appearance = list(default="lhs", rhs="beer"))

inspect(beer\_rules\_rhs)

beer\_rules\_lhs <- apriori(trans,

parameter = list(supp=0.3, conf=0.5,

maxlen=10,

minlen=2),

appearance = list(lhs="beer", default="rhs"))

inspect(beer\_rules\_lhs)

**Output**

> inspect(beer\_rules\_rhs)

lhs rhs support confidence coverage lift count

[1] {apple} => {beer} 0.375 0.75 0.5 1.000000 3

[2] {milk} => {beer} 0.375 0.75 0.5 1.000000 3

[3] {rice} => {beer} 0.500 1.00 0.5 1.333333 4

> inspect(beer\_rules\_lhs)

lhs rhs support confidence coverage lift count

[1] {beer} => {apple} 0.375 0.5000000 0.75 1.000000 3

[2] {beer} => {milk} 0.375 0.5000000 0.75 1.000000 3

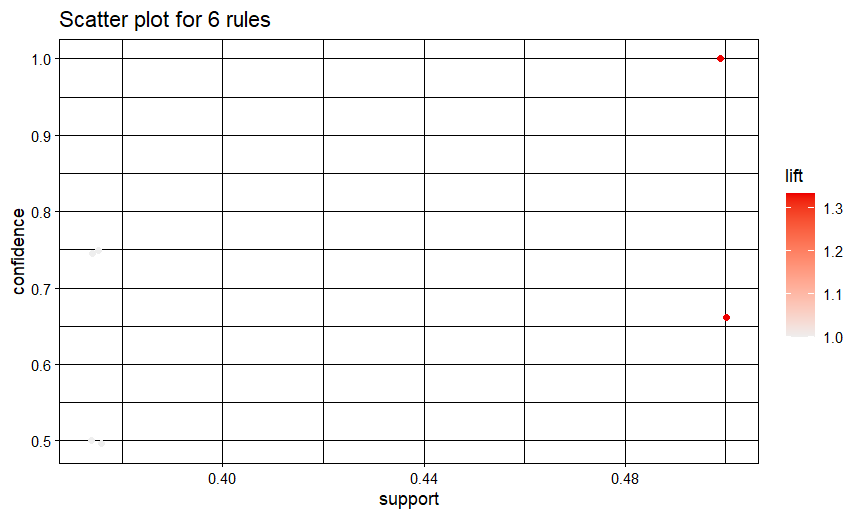
[3] {beer} => {rice} 0.500 0.6666667 0.75 1.333333 4

**Script**

#Visualizing association Rules

plot(rules)

**Output**

****

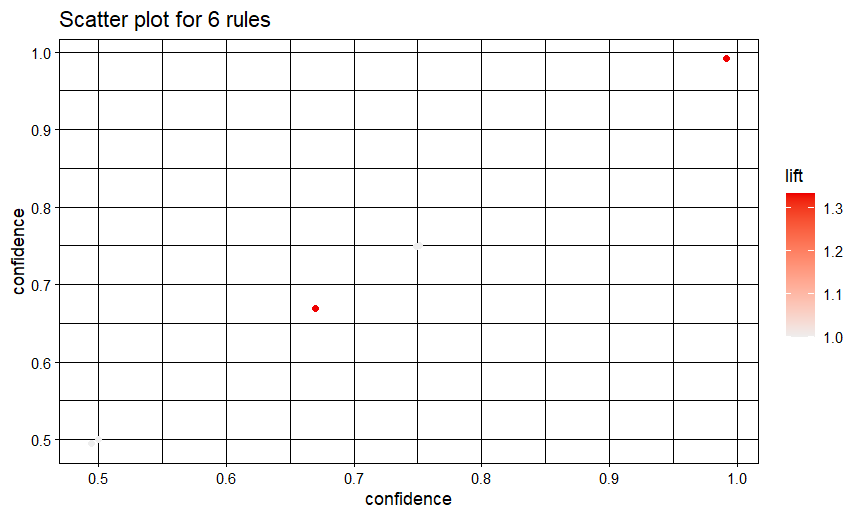
**Script**

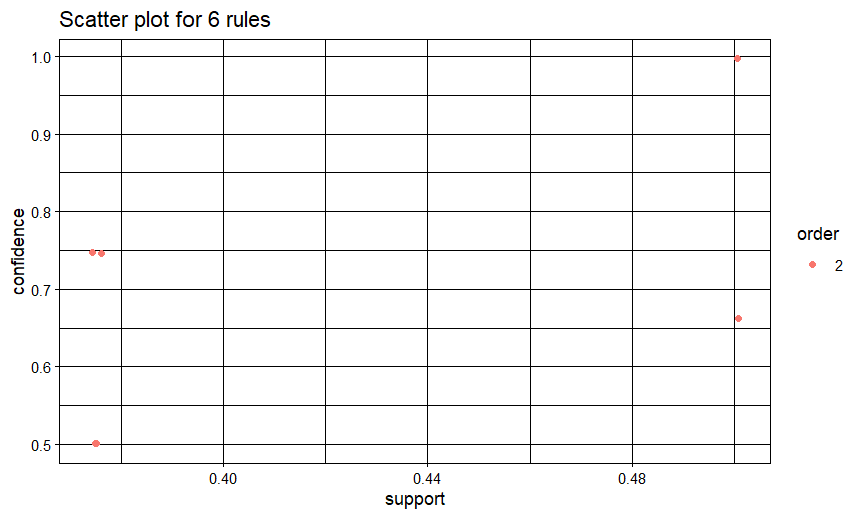
#Confidence as a measure of interest

plot(rules, measure = "confidence")

plot(rules, method = "two-key plot")

**Output**

****

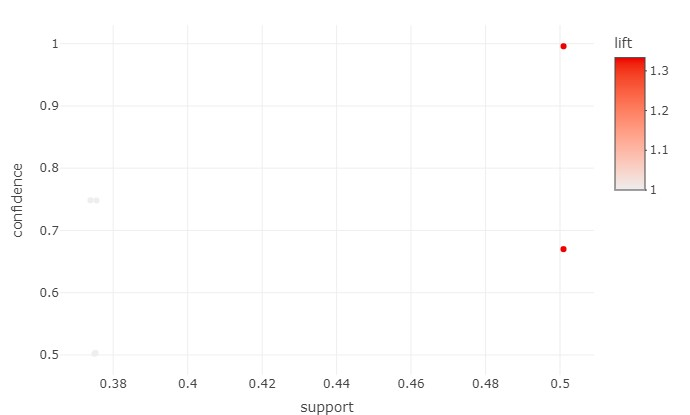
****

**Script**

#Interactive Scatter Plot

plot(rules, engine = "plotly")

**Output**



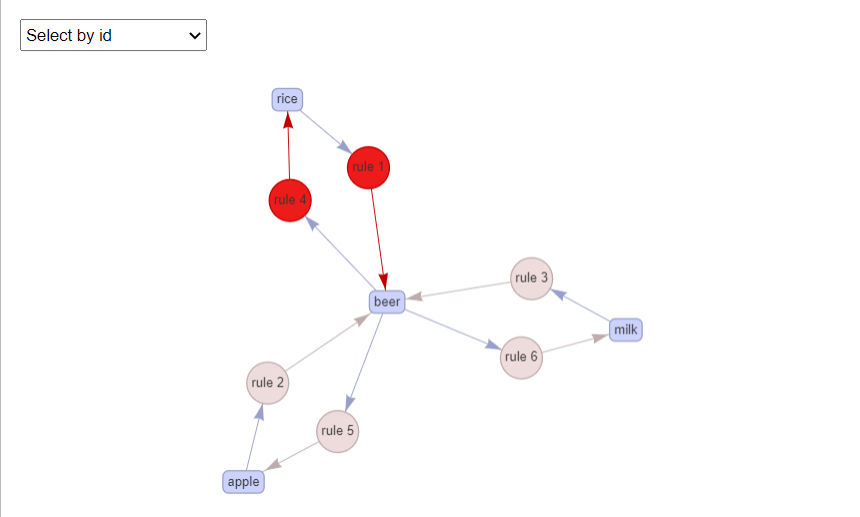
**Script**

#Graph based Visualization

subrules <- head(rules, n = 10, by = "confidence")

plot(subrules, method = "graph", engine = "htmlwidget")

**Output**

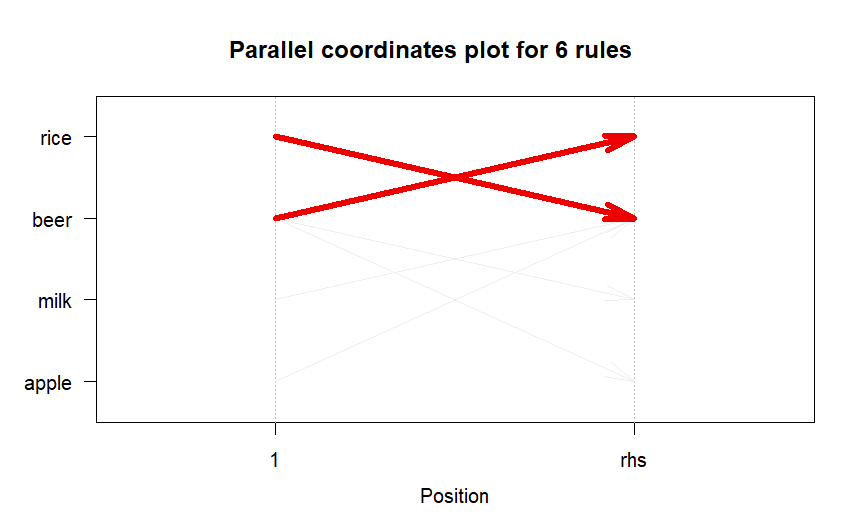
****

**Script**

#Parallel coordinate plot

plot(subrules, method="paracoord")

**Output**

****

**Result**

Thus the Apriori algorithm has been successfully implemented using R programming.