Name: Sanika Deshmukh

Roll – 501872

Batch -2

**Exp- 9 : Association Mining using Apriori**

**Aim:** Implement Apriori algorithm using R.

**Theory:**

The Apriori algorithm uses frequent itemsets to generate association rules, and it is designed to work on the databases that contain transactions. With the help of these association rules, it determines how strongly or how weakly two objects are connected. It is the iterative process for finding the frequent itemsets from the large dataset.

**Steps for Apriori Algorithm :**

1. **Join Step**: This step generates (K+1) itemset from K-itemsets by joining each item with itself.
2. **Prune Step**: This step scans the count of each item in the database. If the candidate item does not meet minimum support, then it is regarded as infrequent and thus it is removed. This step is performed to reduce the size of the candidate itemsets.

Apriori algorithm is a sequence of steps to be followed to find the most frequent itemset in the given database.

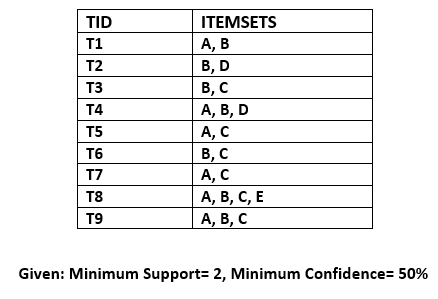
**Step-1:** Determine the support of itemsets in the transactional database, and select the minimum support and confidence.

**Step-2:** Take all supports in the transaction with higher support value than the minimum or selected support value.

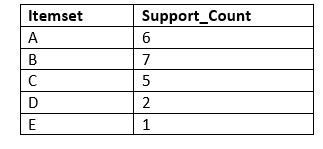
**Step-3:** Find all the rules of these subsets that have higher confidence value than the threshold or minimum confidence.

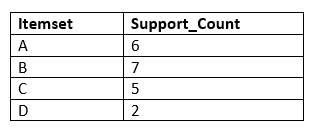
**Step-4:** Sort the rules as the decreasing order of lift.

**EXAMPLE:**

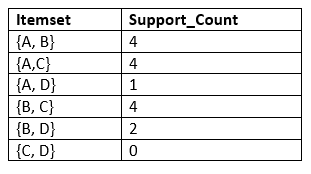


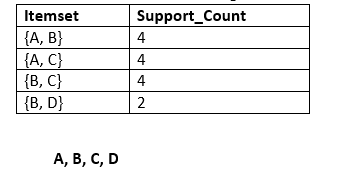
**Step-1: Calculating C1 and L1:**

* In the first step, we will create a table that contains the support count (The frequency of each itemset individually in the dataset) of each itemset in the given dataset. This table is called the Candidate set or C1. ****
* Now, we will take out all the itemsets that have the greater support count that the Minimum Support (2). It will give us the table for the frequent itemset L1.  
  Since all the itemsets have greater or equal support count than the minimum support, except the E, so E itemset will be removed.

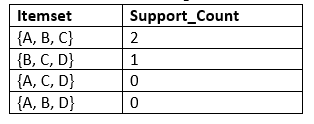
****

**Step-2: Candidate Generation C2, and L2:**

* In this step, we will generate C2 with the help of L1. In C2, we will create the pair of the itemsets of L1 in the form of subsets.
* After creating the subsets, we will again find the support count from the main transaction table of datasets, i.e., how many times these pairs have occurred together in the given dataset. So, we will get the below table for C2: ****
* Again, we need to compare the C2 Support count with the minimum support count, and after comparing, the itemset with less support count will be eliminated from the table C2. It will give us the below table for L2

****

**Step-3: Candidate generation C3, and L3:**

* For C3, we will repeat the same two processes, but now we will form the C3 table with subsets of three itemsets together, and will calculate the support count from the dataset. It will give the below table:  
  
* Now we will create the L3 table. As we can see from the above C3 table, there is only one combination of itemset that has support count equal to the minimum support count. So, the L3 will have only one combination, i.e., {A, B, C}.

**Step-4: Finding the association rules for the subsets:**

To generate the association rules, first, we will create a new table with the possible rules from the occurred combination {A, B.C}. For all the rules, we will calculate the Confidence using formula sup( A ^B)/A. After calculating the confidence value for all rules, we will exclude the rules that have less confidence than the minimum threshold(50%).

Consider the below table:

|  |  |  |
| --- | --- | --- |
| **Rules** | **Support** | **Confidence** |
| A ^B → C | 2 | Sup{(A ^B) ^C}/sup(A ^B)= 2/4=0.5=50% |
| B^C → A | 2 | Sup{(B^C) ^A}/sup(B ^C)= 2/4=0.5=50% |
| A^C → B | 2 | Sup{(A ^C) ^B}/sup(A ^C)= 2/4=0.5=50% |
| C→ A ^B | 2 | Sup{(C^( A ^B)}/sup(C)= 2/5=0.4=40% |
| A→ B^C | 2 | Sup{(A^( B ^C)}/sup(A)= 2/6=0.33=33.33% |
| B→ B^C | 2 | Sup{(B^( B ^C)}/sup(B)= 2/7=0.28=28% |

As the given threshold or minimum confidence is 50%, so the first three rules A ^B → C, B^C → A, and A^C → B can be considered as the strong association rules for the given problem.

**Code:**

**# install arules package (Association Rules)**

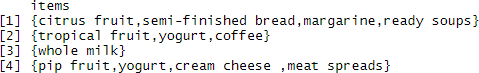
library(arules)

**# install built-in dataset and load it**

library(datasets.load) data("Groceries")

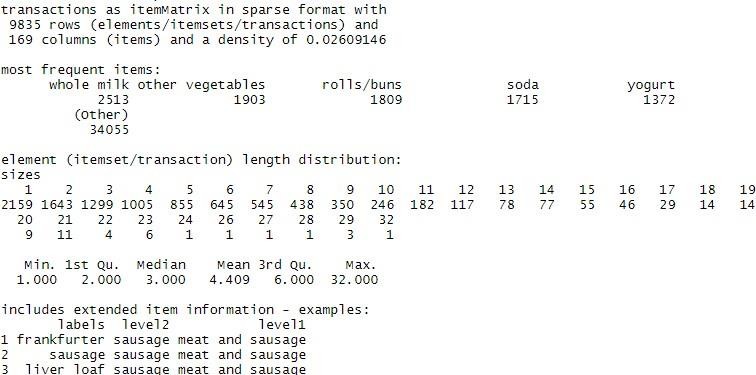
**# Look at the first 4 transactions**

inspect(Groceries[1:4,])



**# See the summary of transaction database**

str(Groceries) summary(Groceries)



**# Examine the frequency percentage of the first item**

itemFrequency(Groceries[,1]) frankfurter 0.05897306

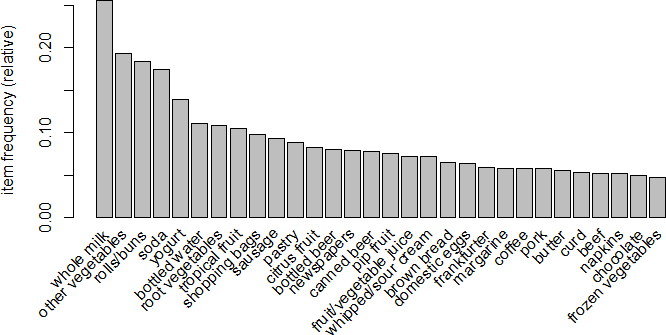
**# Examine the frequency of the first 8 items**

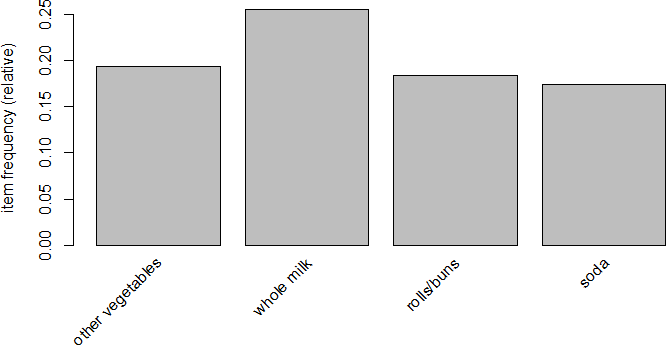
itemFrequency(Groceries[,1:8], type = "absolute")



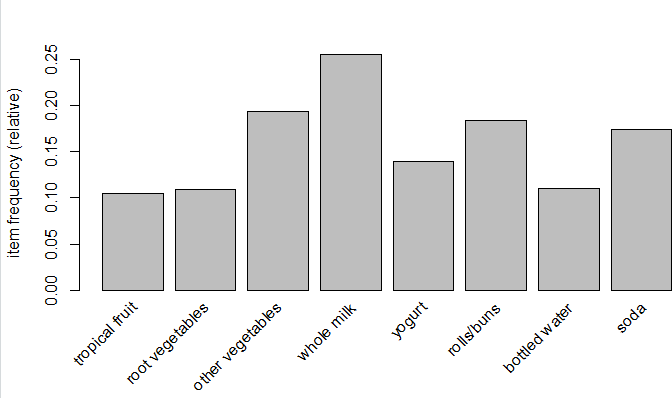
**# Create an item frequency plot for the top 20 items**

itemFrequencyPlot(Groceries, topN=30)



itemFrequencyPlot(Groceries, support=0.15) # for min support = 0.15

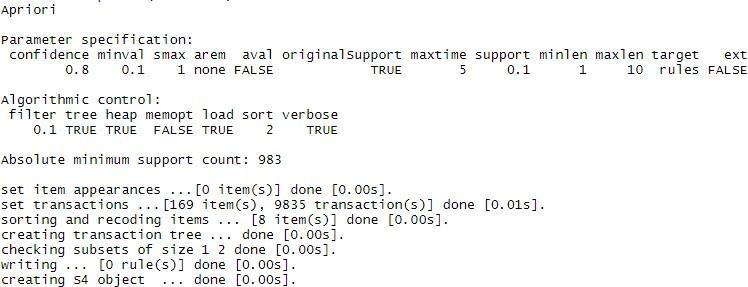
itemFrequencyPlot(Groceries, support=0.10) # for min support = 0.10



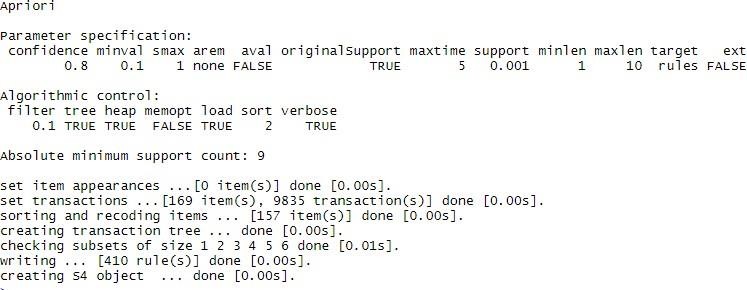
**# Creating rules using Apriori**

**# Default settings result in zero rules learned**

rules <- apriori(Groceries)



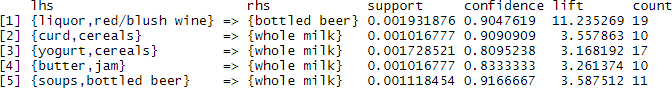
**# Set better support and confidence levels to learn more rule**

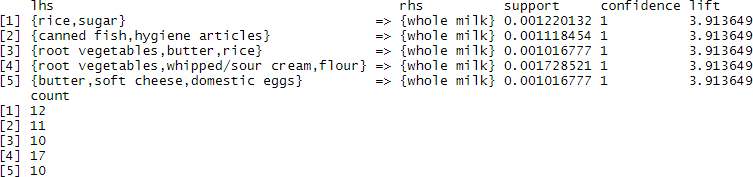
rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8))

rules

set of 410 rules

**# show the top 5 rules**

inspect(rules[1:5])

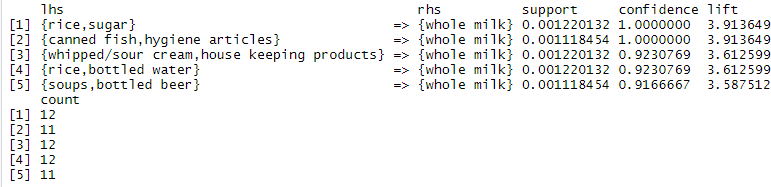
rules <- sort(rules, by = "confidence", decreasing = TRUE) inspect(rules[1:5])

**# Pruning redundant rules**

**# Controlling max length of product in a transaction**

rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.08, maxlen = 3)) rules <- sort(rules, by = "confidence", decreasing = TRUE)

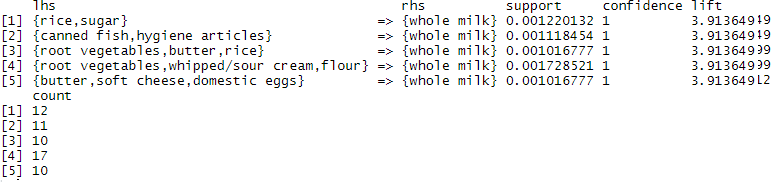
inspect(rules[1:5])



**# Items that lead to whole milk**

rules <- apriori(data = Groceries, parameter = list(supp = 0.001, conf = 0.08), appearance = list(default = "lhs", rhs = "whole milk"), control = list(verbose = F))

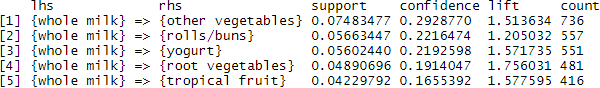
sortedrules <- sort(rules, by = "confidence", decreasing=TRUE) inspect(sortedrules[1:5])



**# Items that are bought with whole milk**

rules <- apriori(data = Groceries, parameter = list(supp = 0.001, conf = 0.15, minlen = 2), appearance = list(default = "rhs", lhs = "whole milk"), control = list(verbose = F))

sortedrules <- sort(rules, by = "confidence", decreasing = TRUE) inspect(sortedrules[1:5])

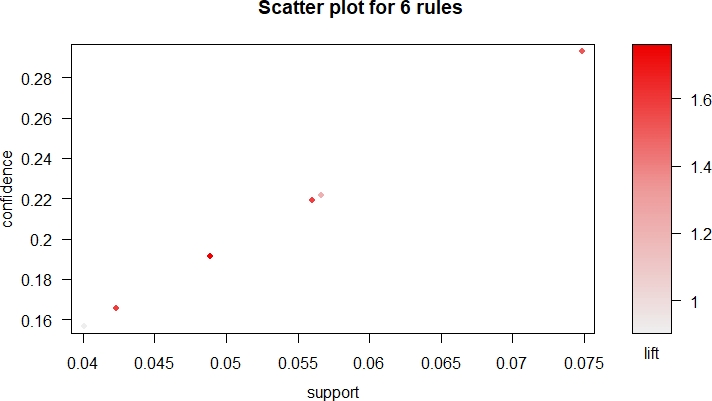


**# Visualization of Rules**

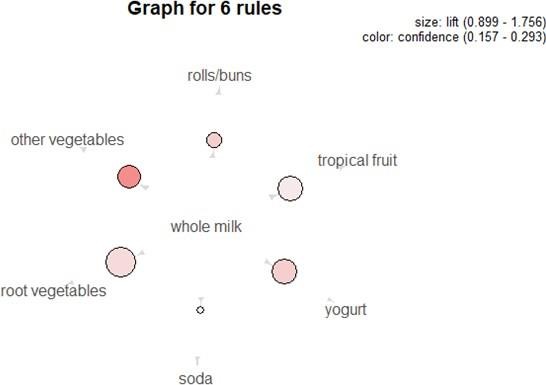
**# Install package arulesViz**

library(arulesViz)

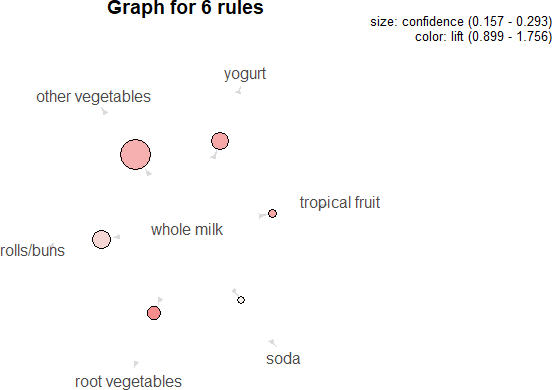
plot(sortedrules)



plot(sortedrules, method = "graph", measure = "lift", shading = "confidence")



plot(sortedrules, method = "graph", measure = "confidence", shading = "lift")



soda\_rule <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.1), appearance = list(default = "rhs", lhs = "soda"))

plot(soda\_rule, method = "grouped")

