Eclat

This notebook includes: \* Applying clat to determine the purchasing relationship between grocery items. \* Visualizing Eclat relationships

# Preprocess data  
#install.packages('arules')  
library(arules)

## Loading required package: Matrix

##   
## Attaching package: 'arules'

## The following objects are masked from 'package:base':  
##   
## abbreviate, write

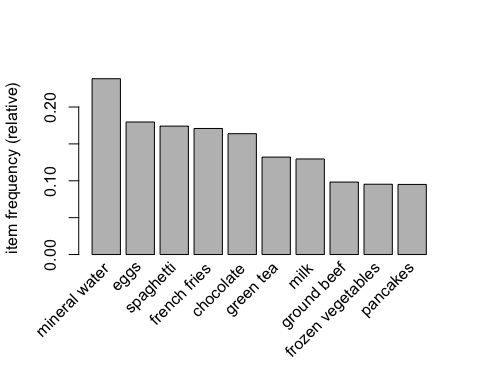
dataset = read.csv('Market\_Basket\_Optimisation.csv')  
dataset = read.transactions('Market\_Basket\_Optimisation.csv', sep = ',', rm.duplicates = TRUE)

## distribution of transactions with duplicates:  
## 1   
## 5

# Explore dataset  
summary(dataset)

## transactions as itemMatrix in sparse format with  
## 7501 rows (elements/itemsets/transactions) and  
## 119 columns (items) and a density of 0.03288973   
##   
## most frequent items:  
## mineral water eggs spaghetti french fries chocolate   
## 1788 1348 1306 1282 1229   
## (Other)   
## 22405   
##   
## element (itemset/transaction) length distribution:  
## sizes  
## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16   
## 1754 1358 1044 816 667 493 391 324 259 139 102 67 40 22 17 4   
## 18 19 20   
## 1 2 1   
##   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.000 2.000 3.000 3.914 5.000 20.000   
##   
## includes extended item information - examples:  
## labels  
## 1 almonds  
## 2 antioxydant juice  
## 3 asparagus

itemFrequencyPlot(dataset, topN = 10)



# Train Eclat on dataset  
rules = eclat(data = dataset, parameter = list(support = 0.003, minlen = 2))

## Eclat  
##   
## parameter specification:  
## tidLists support minlen maxlen target ext  
## FALSE 0.003 2 10 frequent itemsets FALSE  
##   
## algorithmic control:  
## sparse sort verbose  
## 7 -2 TRUE  
##   
## Absolute minimum support count: 22   
##   
## create itemset ...   
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.01s].  
## sorting and recoding items ... [115 item(s)] done [0.00s].  
## creating sparse bit matrix ... [115 row(s), 7501 column(s)] done [0.00s].  
## writing ... [1328 set(s)] done [0.01s].  
## Creating S4 object ... done [0.00s].

# Visualize results  
inspect(sort(rules, by = 'support')[1:10])

## items support count  
## [1] {mineral water,spaghetti} 0.05972537 448   
## [2] {chocolate,mineral water} 0.05265965 395   
## [3] {eggs,mineral water} 0.05092654 382   
## [4] {milk,mineral water} 0.04799360 360   
## [5] {ground beef,mineral water} 0.04092788 307   
## [6] {ground beef,spaghetti} 0.03919477 294   
## [7] {chocolate,spaghetti} 0.03919477 294   
## [8] {eggs,spaghetti} 0.03652846 274   
## [9] {eggs,french fries} 0.03639515 273   
## [10] {frozen vegetables,mineral water} 0.03572857 268