Customer Shopping Prediction using Market Basket Analysis

Adish Joshi

```
install.packages("arules")
```

```
## Installing package into '/home/rstudio-user/R/x86_64-pc-linux-gnu-library/3.6'
## (as 'lib' is unspecified)
```

```
library(arules)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

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

```
install.packages("arulesViz")
```

```
## Installing package into '/home/rstudio-user/R/x86_64-pc-linux-gnu-library/3.6'
## (as 'lib' is unspecified)
```

```
library(arulesViz)
```

```
## Loading required package: grid
```

```
## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus
```

```
data("Groceries")
Groceries
```

```
## transactions in sparse format with
##  9835 transactions (rows) and
##  169 items (columns)
```

```
inspect(Groceries[1:3]) #here we inspect first 3 purcase combinations of the customer.
```

```
##     items
## [1] {citrus fruit,
##      semi-finished bread,
##      margarine,
##      ready soups}
## [2] {tropical fruit,
##      yogurt,
##      coffee}
## [3] {whole milk}
```

```
Q = itemFrequency(Groceries) #Occurence of all the items like milk, soya seeds etc.
W= itemFrequency(Groceries[,1]) #occurence of the first item frankfurtur = 0.059 when we multiply it by
```

```
0.059*9835 # we get 580 which means this item(frankfurture) has occured 580 times among 9835.
```

```
## [1] 580.265
```

```
#But as we are interested in the whole milk let'c how many times the milk has been purchased.
0.2555160142 * 9835
```

```
## [1] 2513
```

```
# we can see that milk has been purchased 2513 times.
itemFrequency(Groceries[,1:6])# provides frequency of first 5 items
```

```
##       frankfurter           sausage        liver loaf               ham
##       0.058973055       0.093950178       0.005083884       0.026029487
##              meat finished products
##       0.025826131       0.006507372
```

```
install.packages('gmodels')
```

```
## Installing package into '/home/rstudio-user/R/x86_64-pc-linux-gnu-library/3.6'
## (as 'lib' is unspecified)
```

```
library('gmodels')

plot(itemFrequency(Groceries), support = 0.10)
```

```
## Warning in plot.window(...): "support" is not a graphical parameter
```

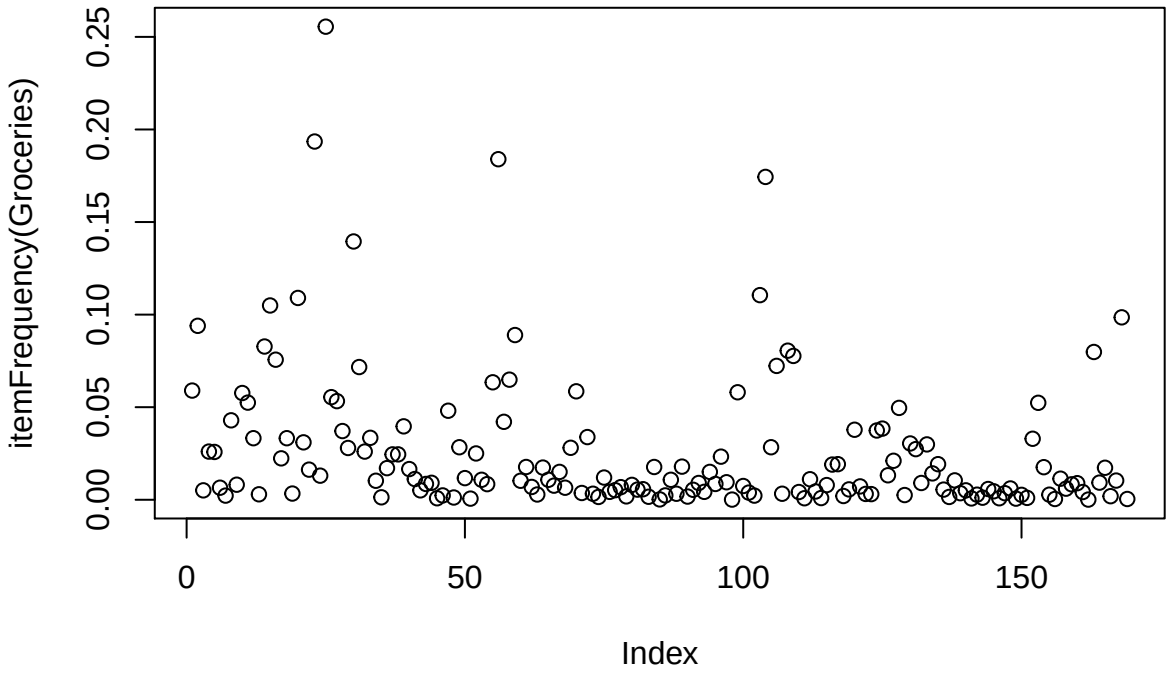
```
## Warning in plot.xy(xy, type, ...): "support" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "support" is not a
## graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "support" is not a
## graphical parameter
```

```
## Warning in box(...): "support" is not a graphical parameter
```

```
## Warning in title(...): "support" is not a graphical parameter
```




```
#Two terms support & confidence are important for the rule mining, where Support = no of item some item
```

```
m1= apriori(Groceries) #we can see that confidence = 0.8 & minval = 0.1
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5     0.1      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 983
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
## sorting and recoding items ... [8 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [0 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
summary(m1)
```

```
## set of 0 rules
```

```
m1= apriori(Groceries,parameter = list(support = 0.007, confidence = 0.25))
```

```
## Apriori
##
```

```
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.25    0.1    1 none FALSE            TRUE       5   0.007      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 68
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [104 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [364 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
summary(m1)
```

```
## set of 364 rules
##
## rule length distribution (lhs + rhs):sizes
##   1   2   3   4
##   1 137 214  12
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   2.651   3.000   4.000
##
## summary of quality measures:
##     support           confidence        coverage            lift
##  Min.   :0.007016   Min.   :0.2500   Min.   :0.01200   Min.   :0.9932
##  1st Qu.:0.008134   1st Qu.:0.2958   1st Qu.:0.02166   1st Qu.:1.6044
##  Median :0.009761   Median :0.3550   Median :0.02888   Median :1.9068
##  Mean   :0.013611   Mean   :0.3740   Mean   :0.03940   Mean   :2.0044
##  3rd Qu.:0.013854   3rd Qu.:0.4417   3rd Qu.:0.04230   3rd Qu.:2.3275
##  Max.   :0.255516   Max.   :0.6389   Max.   :1.00000   Max.   :3.9565
##      count
##  Min.   :  69.0
##  1st Qu.:  80.0
##  Median :  96.0
##  Mean   : 133.9
##  3rd Qu.: 136.2
##  Max.   :2513.0
##
## mining info:
##       data ntransactions support confidence
##  Groceries          9835   0.007       0.25
```

```
#{rule length distribution (lhs + rhs):sizes what this means is for a product A and product B i.e 2 pro
#  1   2   3   4
#  1 137 214  12 }
```

```
inspect(m1[1:6]) # we can see that lift is the factor that a customer lifts the item & puts in his bask
```

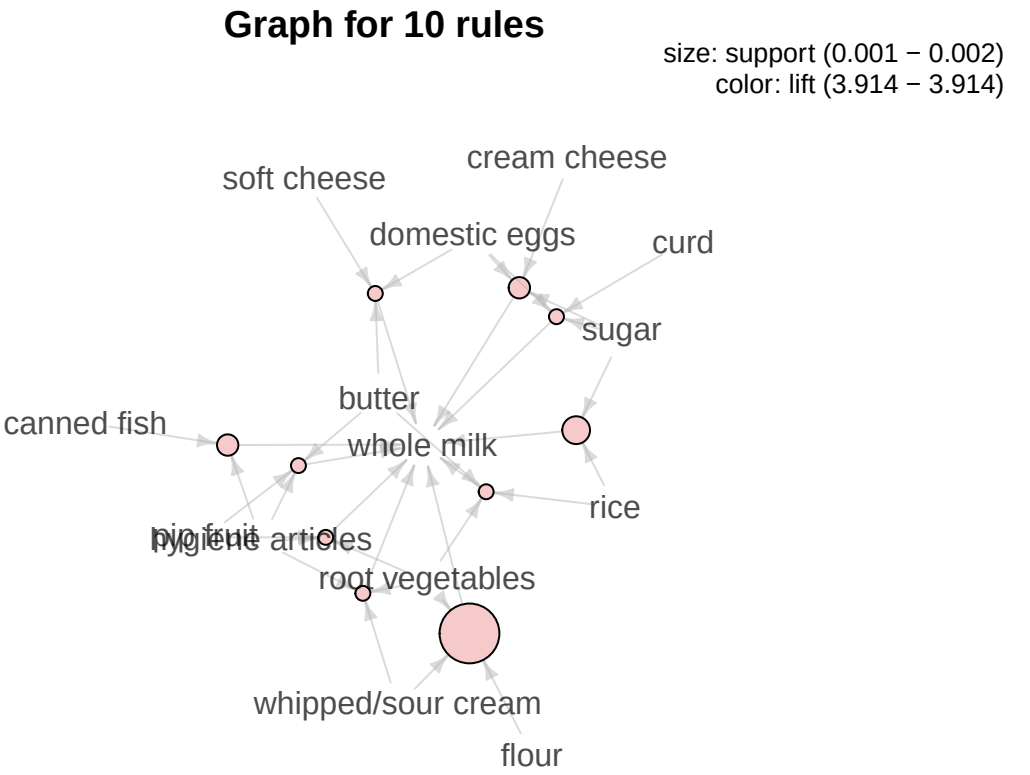
```
##     lhs                     rhs                support     confidence
## [1] {}                   => {whole milk}       0.255516014 0.2555160
## [2] {herbs}              => {root vegetables}  0.007015760 0.4312500
## [3] {herbs}              => {other vegetables} 0.007727504 0.4750000
## [4] {herbs}              => {whole milk}       0.007727504 0.4750000
## [5] {processed cheese}   => {whole milk}       0.007015760 0.4233129
## [6] {semi-finished bread} => {whole milk}      0.007117438 0.4022989
##     coverage   lift     count
## [1] 1.00000000 1.000000 2513
## [2] 0.01626843 3.956477   69
## [3] 0.01626843 2.454874   76
## [4] 0.01626843 1.858983   76
## [5] 0.01657346 1.656698   69
## [6] 0.01769192 1.574457   70
```

```
#LHS the items bought first by the customer & then customer buys the items on rhs, so we can find eithe

m1= apriori(data = Groceries,parameter = list(support = 0.001, confidence = 0.08), appearance = list(de

m1<- sort(m1,by="confidence")
z= m1
#inspect(m1)

plot(m1[1:10],method="graph")
```




```
install.packages("Benchmarking")
```

```
## Installing package into '/home/rstudio-user/R/x86_64-pc-linux-gnu-library/3.6'
```

```
## (as 'lib' is unspecified)
```

```
library(Benchmarking)
```

```
## Loading required package: lpSolveAPI
```

```
## Loading required package: ucminf
```

```
#Efficiency not working if works resend the code to me :)
#efficiency(m1, plot = TRUE, type = "cpD2", thresh = NULL,
    #       shift = 0, amount = NULL)
```