R Notebook

R Programming - Associative Analysis

In this session, we will go through an example of association rules using the arules package. The documentation of this package can be found by visiting the following link: https://www.rdocumentation.org/packages/arules/versions/1.6-4. Below is an extract from its documentation:

"It provides the infrastructure for representing, manipulating and analyzing transaction data and patterns (frequent itemsets and association rules). It also provides interfaces to C implementations of the association mining algorithms Apriori and Eclat."

EXAMPLE

```
# Installing the required arules library
# install.packages("arules")
```

```
# Loading the arules library
library(arules)
```

```
## Warning: package 'arules' was built under R version 4.0.5
```

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

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

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

```
# Loading our transactions dataset from our csv file

# We will use read.transactions fuction which will load data from comma-separated files
# and convert them to the class transactions, which is the kind of data that
# we will require while working with models of association rules

path <- "http://bit.ly/GroceriesDataset"

Transactions <- read.transactions(path, sep = ",")
Transactions
```

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

```r
# Verifying the object's class
# This should show us transactions as the type of data that we will need
class(Transactions)
```

```
## [1] "transactions"
## attr(,"package")
## [1] "arules"
```

```r
# Previewing our first 5 transactions
inspect(Transactions[1:5])
```

```
##     items
## [1] {citrus fruit,
##      margarine,
##      ready soups,
##      semi-finished bread}
## [2] {coffee,
##      tropical fruit,
##      yogurt}
## [3] {whole milk}
## [4] {cream cheese,
##      meat spreads,
##      pip fruit,
##      yogurt}
## [5] {condensed milk,
##      long life bakery product,
##      other vegetables,
##      whole milk}
```

```r
# If we wanted to preview the items that make up our dataset, alternatively we can do the following

items <- as.data.frame(itemLabels(Transactions))
colnames(items) <- "Item"
head(items, 10)
```

```
##                Item
## 1  abrasive cleaner
## 2  artif. sweetener
## 3    baby cosmetics
## 4         baby food
## 5              bags
## 6     baking powder
## 7  bathroom cleaner
## 8              beef
## 9           berries
## 10        beverages
```

```r
# Generating a summary of the transaction dataset

# This would give us some information such as the most purchased items,
# distribution of the item sets (no. of items purchased in each transaction), etc.

summary(Transactions)
```

```
## transactions as itemMatrix in sparse format with
##  9835 rows (elements/itemsets/transactions) and
##  169 columns (items) and a density of 0.02609146
##
## most frequent items:
##       whole milk other vegetables       rolls/buns             soda
##             2513             1903             1809             1715
##           yogurt          (Other)
##             1372            34055
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46
##   17   18   19   20   21   22   23   24   26   27   28   29   32
##   29   14   14    9   11    4    6    1    1    1    1    3    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   4.409   6.000  32.000
##
## includes extended item information - examples:
##             labels
## 1 abrasive cleaner
## 2 artif. sweetener
## 3   baby cosmetics
```

```r
# Exploring the frequency of some articles
# i.e. transacations ranging from 8 to 10 and performing
# some operation in percentage terms of the total transactions

itemFrequency(Transactions[, 8:10],type = "absolute")
```

```
##      beef   berries beverages
##       516       327       256
```

```r
round(itemFrequency(Transactions[, 8:10],type = "relative")*100,2)
```

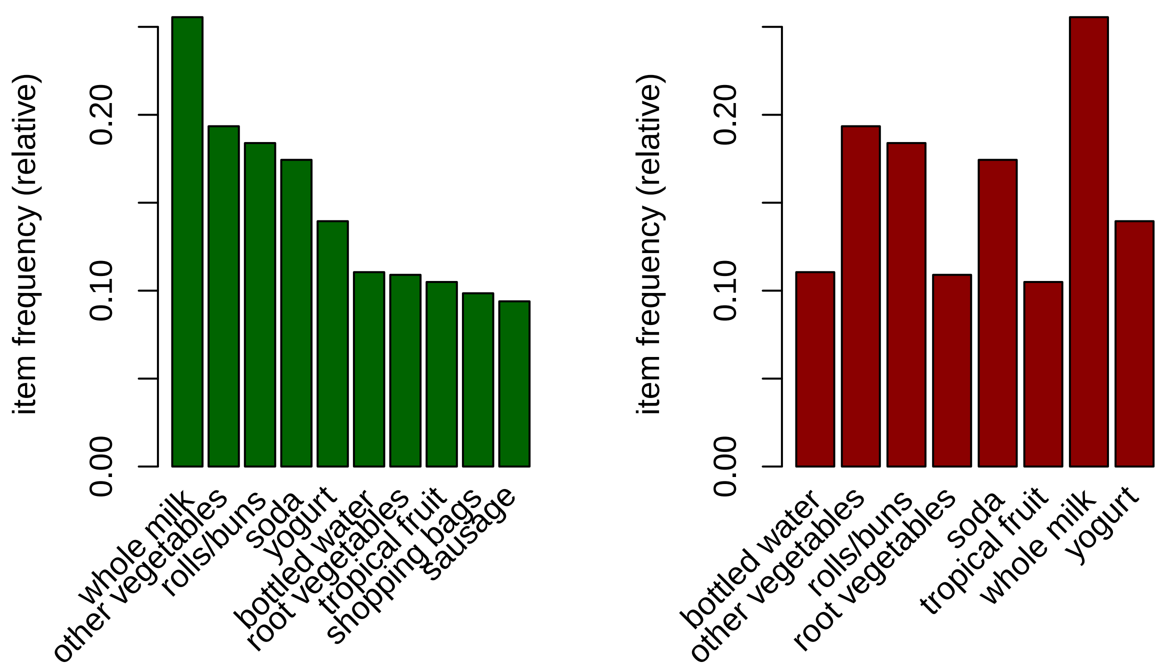
```
##      beef   berries beverages
##      5.25      3.32      2.60
```

```r
# Producing a chart of frequencies and fitering to consider only items with a minimum percentage of sup

# Displaying top 10 most common items in the transactions dataset and the items whose relative importan
```

```r
par(mfrow = c(1, 2))

# plot the frequency of items
itemFrequencyPlot(Transactions, topN = 10, col="darkgreen")
itemFrequencyPlot(Transactions, support = 0.1, col="darkred")
```




```r
# Building a model based on association rules using the apriori function

# We use Min Support as 0.001 and confidence as 0.8

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

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5   0.001      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: 9
```

```
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [157 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.01s].
## writing ... [410 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
rules
```

```
## set of 410 rules
```

- We use measures of significance and interest on the rules, determining which ones are interesting and which to discard
- However, since we built the model using 0.001 Min support and confidence as 0.8 we obtained 410 rules. However, in order to illustrate the sensitivity of the model to these parameters, we will see what happens if we increase the support or lower the confidence level

```r
# Building a apriori model with Min Support as 0.002 and confidence as 0.8.
rules2 <- apriori (Transactions,parameter = list(supp = 0.002, conf = 0.8))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5   0.002      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: 19
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [147 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 done [0.00s].
## writing ... [11 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
# Building apriori model with Min Support as 0.002 and confidence as 0.6.
rules3 <- apriori (Transactions, parameter = list(supp = 0.001, conf = 0.6))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.6    0.1    1 none FALSE            TRUE       5   0.001      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: 9
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [157 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.01s].
## writing ... [2918 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules2
```

```
## set of 11 rules
```

```
rules3
```

```
## set of 2918 rules
```

In our first example, we increased the minimum support of 0.001 to 0.002 and model rules went from 410 to only 11. This would lead us to understand that using a high level of support can make the model lose interesting rules. In the second example, we decreased the minimum confidence level to 0.6 and the number of model rules went from 410 to 2918. This would mean that using a low confidence level increases the number of rules to quite an extent and many will not be useful.

```
# We can perform an exploration of our model through the use of the summary function as shown

summary(rules)
```

```
## set of 410 rules
##
## rule length distribution (lhs + rhs):sizes
##   3   4   5   6
##  29 229 140  12
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   4.000   4.000   4.329   5.000   6.000
##
## summary of quality measures:
##     support           confidence        coverage             lift
##  Min.   :0.001017   Min.   :0.8000   Min.   :0.001017   Min.   : 3.131
##  1st Qu.:0.001017   1st Qu.:0.8333   1st Qu.:0.001220   1st Qu.: 3.312
##  Median :0.001220   Median :0.8462   Median :0.001322   Median : 3.588
##  Mean   :0.001247   Mean   :0.8663   Mean   :0.001449   Mean   : 3.951
##  3rd Qu.:0.001322   3rd Qu.:0.9091   3rd Qu.:0.001627   3rd Qu.: 4.341
##  Max.   :0.003152   Max.   :1.0000   Max.   :0.003559   Max.   :11.235
```

```
##      count
##  Min.   :10.00
##  1st Qu.:10.00
##  Median :12.00
##  Mean   :12.27
##  3rd Qu.:13.00
##  Max.   :31.00
##
## mining info:
##          data ntransactions support confidence
##  Transactions          9835   0.001        0.8
```

```r
# Observing rules built in our model i.e. first 5 model rules
inspect(rules[1:5])
```

```
##     lhs                          rhs            support     confidence
## [1] {liquor,red/blush wine} => {bottled beer} 0.001931876 0.9047619
## [2] {cereals,curd}          => {whole milk}   0.001016777 0.9090909
## [3] {cereals,yogurt}        => {whole milk}   0.001728521 0.8095238
## [4] {butter,jam}            => {whole milk}   0.001016777 0.8333333
## [5] {bottled beer,soups}    => {whole milk}   0.001118454 0.9166667
##     coverage    lift      count
## [1] 0.002135231 11.235269 19
## [2] 0.001118454  3.557863 10
## [3] 0.002135231  3.168192 17
## [4] 0.001220132  3.261374 10
## [5] 0.001220132  3.587512 11
```

```r
# Interpretation of the first rule:
# ---
# If someone buys liquor and red/blush wine, they are 90% likely to buy bottled beer too
# ---
```

```r
# Ordering these rules by a criteria such as the level of confidence then looking at the first five rul
# We can also use different criteria such as: (by = "lift" or by = "support")

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

```
##     lhs                        rhs              support confidence    coverage     lift count
## [1] {rice,
##      sugar}               => {whole milk} 0.001220132          1 0.001220132 3.913649    12
## [2] {canned fish,
##      hygiene articles}    => {whole milk} 0.001118454          1 0.001118454 3.913649    11
## [3] {butter,
##      rice,
##      root vegetables}     => {whole milk} 0.001016777          1 0.001016777 3.913649    10
## [4] {flour,
##      root vegetables,
##      whipped/sour cream} => {whole milk} 0.001728521          1 0.001728521 3.913649    17
## [5] {butter,
##      domestic eggs,
##      soft cheese}        => {whole milk} 0.001016777          1 0.001016777 3.913649    10
```

```r
# Interpretation
# ---
# The given five rules have a confidence of 100
# ---
```

```r
# If we're interested in making a promotion relating to the sale of yogurt, we could create a subset of

# This would tell us the items that the customers bought before purchasing yogurt

yogurt <- subset(rules, subset = rhs %pin% "yogurt")

# Then order by confidence
yogurt <- sort(yogurt, by="confidence", decreasing=TRUE)
inspect(yogurt[1:5])
```

```
##     lhs                        rhs          support confidence    coverage     lift count
## [1] {butter,
##      cream cheese,
##      root vegetables}    => {yogurt} 0.001016777  0.9090909 0.001118454 6.516698    10
## [2] {butter,
##      sliced cheese,
##      tropical fruit,
##      whole milk}         => {yogurt} 0.001016777  0.9090909 0.001118454 6.516698    10
## [3] {cream cheese,
##      curd,
##      other vegetables,
##      whipped/sour cream} => {yogurt} 0.001016777  0.9090909 0.001118454 6.516698    10
## [4] {butter,
##      other vegetables,
##      tropical fruit,
##      white bread}        => {yogurt} 0.001016777  0.9090909 0.001118454 6.516698    10
## [5] {pip fruit,
##      sausage,
##      sliced cheese}      => {yogurt} 0.001220132  0.8571429 0.001423488 6.144315    12
```

```r
# What if we wanted to determine items that customers might buy who have previously bought yogurt?

# Subset the rules
yogurt <- subset(rules, subset = lhs %pin% "yogurt")

# Order by confidence
yogurt <- sort(yogurt, by="confidence", decreasing=TRUE)

# inspect top 5
inspect(yogurt[15:19])
```

```
##     lhs                        rhs              support confidence    coverage     lift count
## [1] {butter,
##      domestic eggs,
##      tropical fruit,
##      yogurt}             => {whole milk} 0.001220132  0.9230769 0.001321810 3.612599    12
## [2] {cream cheese,
```

```
##      other vegetables,
##      pip fruit,
##      yogurt}              => {whole milk} 0.001118454  0.9166667 0.001220132 3.587512    11
## [3] {curd,
##      domestic eggs,
##      tropical fruit,
##      yogurt}              => {whole milk} 0.001118454  0.9166667 0.001220132 3.587512    11
## [4] {butter,
##      domestic eggs,
##      root vegetables,
##      yogurt}              => {whole milk} 0.001118454  0.9166667 0.001220132 3.587512    11
## [5] {domestic eggs,
##      tropical fruit,
##      whipped/sour cream,
##      yogurt}              => {whole milk} 0.001118454  0.9166667 0.001220132 3.587512    11
```

CHALLENGES

CHALLENGE 1

```r
# Question: Build an apriori model previewing the rules with the highest confidence interval given the

url = "http://bit.ly/AssociativeAnalysisDataset"

# Loading the dataset
Transactions = read.transactions(url, sep = ",")
```

```
## Warning in asMethod(object): removing duplicated items in transactions
```

```r
Transactions
```

```
## transactions in sparse format with
##  7501 transactions (rows) and
##  119 items (columns)
```

```r
# Inspecting the object type
class(Transactions)
```

```
## [1] "transactions"
## attr(,"package")
## [1] "arules"
```

```r
# Previewing the first 5 transactions
inspect(Transactions[1:5])
```

```
##     items
## [1] {almonds,
##      antioxydant juice,
##      avocado,
```

```
##      cottage cheese,
##      energy drink,
##      frozen smoothie,
##      green grapes,
##      green tea,
##      honey,
##      low fat yogurt,
##      mineral water,
##      olive oil,
##      salad,
##      salmon,
##      shrimp,
##      spinach,
##      tomato juice,
##      vegetables mix,
##      whole weat flour,
##      yams}
## [2] {burgers,
##      eggs,
##      meatballs}
## [3] {chutney}
## [4] {avocado,
##      turkey}
## [5] {energy bar,
##      green tea,
##      milk,
##      mineral water,
##      whole wheat rice}
```

```r
# Alternatively
items <- as.data.frame(itemLabels(Transactions))
colnames(items) <- "Item"
head(items)
```

```
##                Item
## 1           almonds
## 2 antioxydant juice
## 3         asparagus
## 4           avocado
## 5       babies food
## 6             bacon
```

```r
# Summary of the transactions
summary(Transactions)
```

```
## 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
```

```r
# Frequency of some articles
itemFrequency(Transactions[,10:15], type = "absolute")
```

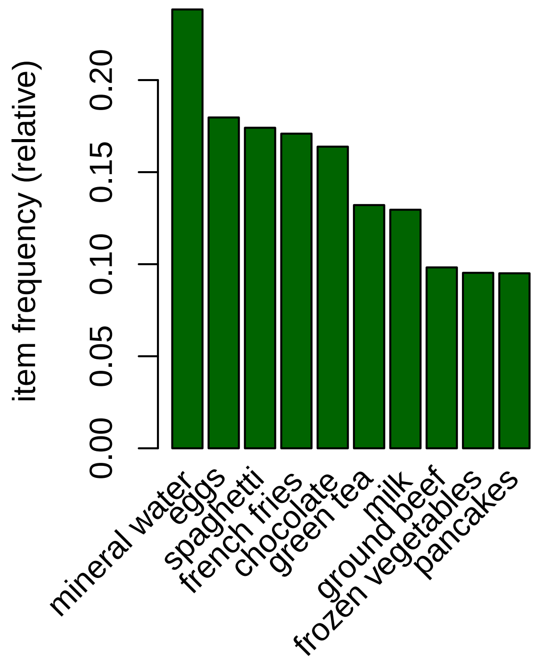
```
##   body spray      bramble     brownies    bug spray burger sauce      burgers
##           86           14          253           65           44          654
```

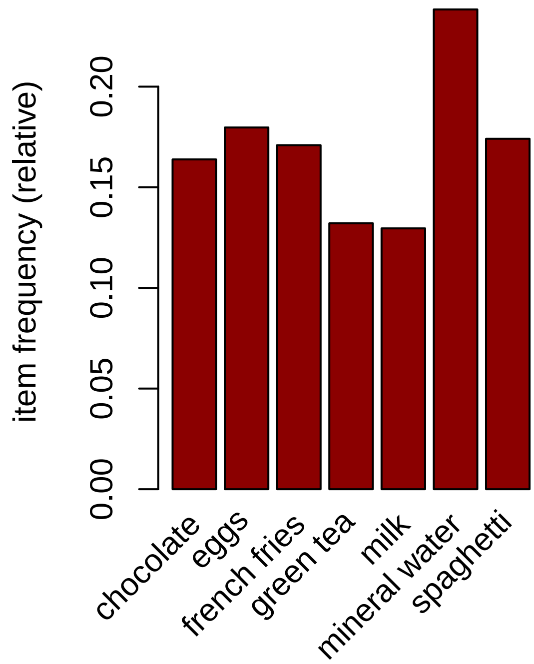
```r
round(itemFrequency(Transactions[,10:15], type = "relative")*100,2)
```

```
##   body spray      bramble     brownies    bug spray burger sauce      burgers
##         1.15         0.19         3.37         0.87         0.59         8.72
```

```r
# Displaying top 10 most common items with at least 10% relative importance
par(mfrow = c(1,2))

# Plots
itemFrequencyPlot(Transactions, topN = 10, col = "darkgreen")
itemFrequencyPlot(Transactions, support = 0.1, col = "darkred")
```






```r
# Building the model
rules <- apriori(Transactions, parameter = list(supp = 0.001, conf = 0.75))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.75    0.1    1 none FALSE            TRUE       5   0.001      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: 7
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.00s].
## sorting and recoding items ... [116 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.00s].
## writing ... [110 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
inspect(sort(rules, by = "confidence"))
```

```
##       lhs                         rhs                 support confidence    coverage      lift cou
## [1]   {french fries,
##        mushroom cream sauce,
##        pasta}                  => {escalope}      0.001066524  1.0000000 0.001066524 12.606723
## [2]   {ground beef,
##        light cream,
##        olive oil}              => {mineral water} 0.001199840  1.0000000 0.001199840  4.195190
## [3]   {cake,
##        meatballs,
##        mineral water}          => {milk}          0.001066524  1.0000000 0.001066524  7.717078
## [4]   {cake,
##        olive oil,
##        shrimp}                 => {mineral water} 0.001199840  1.0000000 0.001199840  4.195190
## [5]   {mushroom cream sauce,
##        pasta}                  => {escalope}      0.002532996  0.9500000 0.002666311 11.976387
## [6]   {red wine,
##        soup}                   => {mineral water} 0.001866418  0.9333333 0.001999733  3.915511
## [7]   {eggs,
##        mineral water,
##        pasta}                  => {shrimp}        0.001333156  0.9090909 0.001466471 12.722185
## [8]   {herb & pepper,
##        mineral water,
##        rice}                   => {ground beef}   0.001333156  0.9090909 0.001466471  9.252498
## [9]   {ground beef,
##        pancakes,
##        whole wheat rice}       => {mineral water} 0.001333156  0.9090909 0.001466471  3.813809
## [10]  {frozen vegetables,
##        milk,
##        spaghetti,
##        turkey}                 => {mineral water} 0.001199840  0.9000000 0.001333156  3.775671
## [11]  {chocolate,
##        frozen vegetables,
##        olive oil,
##        shrimp}                 => {mineral water} 0.001199840  0.9000000 0.001333156  3.775671
## [12]  {frozen smoothie,
##        spinach}                => {mineral water} 0.001066524  0.8888889 0.001199840  3.729058
## [13]  {black tea,
##        spaghetti,
##        turkey}                 => {eggs}          0.001066524  0.8888889 0.001199840  4.946258
## [14]  {light cream,
##        mineral water,
##        shrimp}                 => {spaghetti}     0.001066524  0.8888889 0.001199840  5.105326
## [15]  {cake,
##        meatballs,
##        milk}                   => {mineral water} 0.001066524  0.8888889 0.001199840  3.729058
## [16]  {grated cheese,
##        mineral water,
##        rice}                   => {ground beef}   0.001066524  0.8888889 0.001199840  9.046887
## [17]  {cake,
##        olive oil,
##        whole wheat pasta}      => {mineral water} 0.001066524  0.8888889 0.001199840  3.729058
```

```
## [18]  {escalope,
##        hot dogs,
##        mineral water}        => {milk}              0.001066524  0.8888889 0.001199840  6.859625
## [19]  {brownies,
##        eggs,
##        ground beef}          => {mineral water}     0.001066524  0.8888889 0.001199840  3.729058
## [20]  {chicken,
##        fresh bread,
##        pancakes}             => {mineral water}     0.001066524  0.8888889 0.001199840  3.729058
## [21]  {ground beef,
##        salmon,
##        shrimp}               => {spaghetti}         0.001066524  0.8888889 0.001199840  5.105326
## [22]  {burgers,
##        milk,
##        salmon}               => {spaghetti}         0.001066524  0.8888889 0.001199840  5.105326
## [23]  {chocolate,
##        soup,
##        turkey}               => {mineral water}     0.001066524  0.8888889 0.001199840  3.729058
## [24]  {escalope,
##        french fries,
##        shrimp}               => {chocolate}         0.001066524  0.8888889 0.001199840  5.425188
## [25]  {chocolate,
##        ground beef,
##        milk,
##        mineral water,
##        spaghetti}            => {frozen vegetables} 0.001066524  0.8888889 0.001199840  9.325253
## [26]  {frozen vegetables,
##        ground beef,
##        mineral water,
##        shrimp}               => {spaghetti}         0.001733102  0.8666667 0.001999733  4.977693
## [27]  {chocolate,
##        frozen vegetables,
##        shrimp,
##        spaghetti}            => {mineral water}     0.001733102  0.8666667 0.001999733  3.635831
## [28]  {ground beef,
##        nonfat milk}          => {mineral water}     0.001599787  0.8571429 0.001866418  3.595877
## [29]  {milk,
##        pasta}                => {shrimp}            0.001599787  0.8571429 0.001866418 11.995203
## [30]  {turkey,
##        whole wheat pasta}    => {mineral water}     0.001466471  0.8461538 0.001733102  3.549776
## [31]  {burgers,
##        frozen vegetables,
##        pancakes}             => {spaghetti}         0.001466471  0.8461538 0.001733102  4.859877
## [32]  {frozen vegetables,
##        milk,
##        shrimp,
##        spaghetti}            => {mineral water}     0.001466471  0.8461538 0.001733102  3.549776
## [33]  {chocolate,
##        eggs,
##        frozen vegetables,
##        ground beef}          => {mineral water}     0.001466471  0.8461538 0.001733102  3.549776
## [34]  {frozen vegetables,
##        olive oil,
##        tomatoes}             => {spaghetti}         0.002133049  0.8421053 0.002532996  4.836624
```

```
## [35] {meatballs,
##       whole wheat pasta}    => {milk}            0.001333156  0.8333333 0.001599787  6.430898
## [36] {mineral water,
##       pasta,
##       shrimp}               => {eggs}            0.001333156  0.8333333 0.001599787  4.637117
## [37] {green tea,
##       ground beef,
##       tomato sauce}         => {spaghetti}       0.001333156  0.8333333 0.001599787  4.786243
## [38] {olive oil,
##       soup,
##       tomatoes}             => {mineral water}   0.001333156  0.8333333 0.001599787  3.495992
## [39] {frozen vegetables,
##       tomatoes,
##       whole wheat rice}     => {spaghetti}       0.001333156  0.8333333 0.001599787  4.786243
## [40] {frozen vegetables,
##       olive oil,
##       shrimp}               => {mineral water}   0.001866418  0.8235294 0.002266364  3.454862
## [41] {nonfat milk,
##       turkey}               => {mineral water}   0.001199840  0.8181818 0.001466471  3.432428
## [42] {cooking oil,
##       fromage blanc}        => {mineral water}   0.001199840  0.8181818 0.001466471  3.432428
## [43] {black tea,
##       frozen smoothie}      => {milk}            0.001199840  0.8181818 0.001466471  6.313973
## [44] {chicken,
##       protein bar}          => {spaghetti}       0.001199840  0.8181818 0.001466471  4.699220
## [45] {french fries,
##       herb & pepper,
##       milk}                 => {mineral water}   0.001199840  0.8181818 0.001466471  3.432428
## [46] {burgers,
##       frozen vegetables,
##       olive oil}            => {mineral water}   0.001199840  0.8181818 0.001466471  3.432428
## [47] {frozen vegetables,
##       milk,
##       olive oil,
##       soup}                 => {mineral water}   0.001199840  0.8181818 0.001466471  3.432428
## [48] {frozen vegetables,
##       ground beef,
##       mineral water,
##       tomatoes}             => {spaghetti}       0.001199840  0.8181818 0.001466471  4.699220
## [49] {chocolate,
##       eggs,
##       olive oil,
##       spaghetti}            => {mineral water}   0.001199840  0.8181818 0.001466471  3.432428
## [50] {chocolate,
##       milk,
##       shrimp,
##       spaghetti}            => {mineral water}   0.001199840  0.8181818 0.001466471  3.432428
## [51] {bacon,
##       pancakes}             => {spaghetti}       0.001733102  0.8125000 0.002133049  4.666587
## [52] {frozen vegetables,
##       olive oil,
##       soup}                 => {mineral water}   0.001733102  0.8125000 0.002133049  3.408592
## [53] {black tea,
##       salmon}               => {mineral water}   0.001066524  0.8000000 0.001333156  3.356152
```

```
## [54]  {red wine,
##        tomato sauce}         => {chocolate}         0.001066524  0.8000000 0.001333156  4.882669
## [55]  {pancakes,
##        tomato sauce}         => {mineral water}     0.001066524  0.8000000 0.001333156  3.356152
## [56]  {milk,
##        spaghetti,
##        strong cheese}        => {mineral water}     0.001066524  0.8000000 0.001333156  3.356152
## [57]  {grated cheese,
##        ground beef,
##        rice}                 => {mineral water}     0.001066524  0.8000000 0.001333156  3.356152
## [58]  {milk,
##        mineral water,
##        parmesan cheese}      => {spaghetti}         0.001066524  0.8000000 0.001333156  4.594793
## [59]  {oil,
##        shrimp,
##        spaghetti}            => {mineral water}     0.001066524  0.8000000 0.001333156  3.356152
## [60]  {cooking oil,
##        mineral water,
##        red wine}             => {spaghetti}         0.001066524  0.8000000 0.001333156  4.594793
## [61]  {escalope,
##        hot dogs,
##        milk}                 => {mineral water}     0.001066524  0.8000000 0.001333156  3.356152
## [62]  {chocolate,
##        hot dogs,
##        milk}                 => {mineral water}     0.001066524  0.8000000 0.001333156  3.356152
## [63]  {avocado,
##        burgers,
##        milk}                 => {spaghetti}         0.001066524  0.8000000 0.001333156  4.594793
## [64]  {cookies,
##        green tea,
##        milk}                 => {french fries}      0.001066524  0.8000000 0.001333156  4.680811
## [65]  {chocolate,
##        olive oil,
##        soup}                 => {mineral water}     0.001599787  0.8000000 0.001999733  3.356152
## [66]  {cooking oil,
##        eggs,
##        olive oil}            => {mineral water}     0.001066524  0.8000000 0.001333156  3.356152
## [67]  {burgers,
##        frozen vegetables,
##        low fat yogurt}       => {mineral water}     0.001066524  0.8000000 0.001333156  3.356152
## [68]  {burgers,
##        ground beef,
##        olive oil}            => {milk}              0.001066524  0.8000000 0.001333156  6.173663
## [69]  {cake,
##        eggs,
##        milk,
##        turkey}               => {mineral water}     0.001066524  0.8000000 0.001333156  3.356152
## [70]  {frozen vegetables,
##        mineral water,
##        olive oil,
##        tomatoes}             => {spaghetti}         0.001066524  0.8000000 0.001333156  4.594793
## [71]  {chocolate,
##        eggs,
##        milk,
```

```
##         olive oil}           => {mineral water}     0.001066524  0.8000000 0.001333156  3.356152
## [72]  {chocolate,
##         french fries,
##         mineral water,
##         olive oil}           => {spaghetti}         0.001066524  0.8000000 0.001333156  4.594793
## [73]  {chocolate,
##         frozen vegetables,
##         pancakes,
##         shrimp}              => {mineral water}     0.001066524  0.8000000 0.001333156  3.356152
## [74]  {french fries,
##         milk,
##         pancakes,
##         spaghetti}           => {mineral water}     0.001066524  0.8000000 0.001333156  3.356152
## [75]  {hot dogs,
##         olive oil,
##         spaghetti}           => {mineral water}     0.001466471  0.7857143 0.001866418  3.296221
## [76]  {eggs,
##         olive oil,
##         soup}                => {mineral water}     0.001466471  0.7857143 0.001866418  3.296221
## [77]  {cake,
##         milk,
##         soup}                => {mineral water}     0.001466471  0.7857143 0.001866418  3.296221
## [78]  {green tea,
##         olive oil,
##         tomatoes}            => {spaghetti}         0.001466471  0.7857143 0.001866418  4.512743
## [79]  {pancakes,
##         soup,
##         spaghetti}           => {mineral water}     0.002266364  0.7727273 0.002932942  3.241738
## [80]  {barbecue sauce,
##         chocolate}           => {mineral water}     0.001333156  0.7692308 0.001733102  3.227069
## [81]  {cottage cheese,
##         milk,
##         spaghetti}           => {mineral water}     0.001333156  0.7692308 0.001733102  3.227069
## [82]  {burgers,
##         herb & pepper,
##         spaghetti}           => {ground beef}       0.001333156  0.7692308 0.001733102  7.829037
## [83]  {chocolate,
##         cooking oil,
##         frozen vegetables}   => {milk}              0.001333156  0.7692308 0.001733102  5.936214
## [84]  {chicken,
##         frozen vegetables,
##         olive oil}           => {spaghetti}         0.001333156  0.7692308 0.001733102  4.418070
## [85]  {frozen vegetables,
##         green tea,
##         olive oil}           => {spaghetti}         0.001333156  0.7692308 0.001733102  4.418070
## [86]  {chocolate,
##         olive oil,
##         pancakes,
##         spaghetti}           => {mineral water}     0.001333156  0.7692308 0.001733102  3.227069
## [87]  {chocolate,
##         mineral water,
##         olive oil,
##         pancakes}            => {spaghetti}         0.001333156  0.7692308 0.001733102  4.418070
## [88]  {frozen vegetables,
```

```
##        milk,
##        soup}                  => {mineral water}     0.003066258  0.7666667 0.003999467  3.216312     2
## [89]  {cereals,
##        ground beef,
##        mineral water}         => {spaghetti}         0.001733102  0.7647059 0.002266364  4.392082     1
## [90]  {burgers,
##        frozen vegetables,
##        ground beef}           => {mineral water}     0.001733102  0.7647059 0.002266364  3.208087     1
## [91]  {blueberries,
##        eggs}                  => {mineral water}     0.001599787  0.7500000 0.002133049  3.146393     1
## [92]  {mineral water,
##        pasta}                 => {shrimp}            0.001599787  0.7500000 0.002133049 10.495802     1
## [93]  {shrimp,
##        tomato sauce}          => {spaghetti}         0.001199840  0.7500000 0.001599787  4.307619
## [94]  {frozen vegetables,
##        milk,
##        parmesan cheese}       => {spaghetti}         0.001199840  0.7500000 0.001599787  4.307619
## [95]  {cereals,
##        french fries,
##        milk}                  => {mineral water}     0.001199840  0.7500000 0.001599787  3.146393
## [96]  {escalope,
##        milk,
##        salmon}                => {mineral water}     0.001199840  0.7500000 0.001599787  3.146393
## [97]  {chocolate,
##        herb & pepper,
##        pancakes}              => {mineral water}     0.001199840  0.7500000 0.001599787  3.146393
## [98]  {chocolate,
##        soup,
##        tomatoes}              => {mineral water}     0.001199840  0.7500000 0.001599787  3.146393
## [99]  {frozen vegetables,
##        tomatoes,
##        whole wheat rice}      => {mineral water}     0.001199840  0.7500000 0.001599787  3.146393
## [100] {mineral water,
##        tomatoes,
##        turkey}                => {spaghetti}         0.001599787  0.7500000 0.002133049  4.307619     1
## [101] {frozen vegetables,
##        milk,
##        turkey}                => {mineral water}     0.001999733  0.7500000 0.002666311  3.146393     1
## [102] {cake,
##        chicken,
##        milk}                  => {mineral water}     0.001599787  0.7500000 0.002133049  3.146393     1
## [103] {frozen smoothie,
##        ground beef,
##        pancakes}              => {spaghetti}         0.001199840  0.7500000 0.001599787  4.307619
## [104] {burgers,
##        olive oil,
##        pancakes}              => {chocolate}         0.001199840  0.7500000 0.001599787  4.577502
## [105] {burgers,
##        olive oil,
##        pancakes}              => {spaghetti}         0.001199840  0.7500000 0.001599787  4.307619
## [106] {frozen vegetables,
##        ground beef,
##        shrimp}                => {spaghetti}         0.002399680  0.7500000 0.003199573  4.307619     1
## [107] {frozen vegetables,
```

```
##        ground beef,
##        herb & pepper,
##        spaghetti}             => {mineral water}     0.001199840  0.7500000 0.001599787  3.146393
## [108] {chocolate,
##        olive oil,
##        shrimp,
##        spaghetti}             => {mineral water}     0.001199840  0.7500000 0.001599787  3.146393
## [109] {eggs,
##        frozen vegetables,
##        milk,
##        olive oil}             => {mineral water}     0.001199840  0.7500000 0.001599787  3.146393
## [110] {chocolate,
##        frozen vegetables,
##        ground beef,
##        milk}                  => {mineral water}     0.001999733  0.7500000 0.002666311  3.146393
```

- 110 rules have been created

CHALLENGE 2

```
# Question: Build an apriori model previewing the rules with the highest confidence interval given the
```

```
# No dataset
```