Association Analysis

2022-04-01

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

```
## Warning: package 'arules' was built under R version 4.1.3
```

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

```
## Warning: package 'Matrix' was built under R version 4.1.3
```

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

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

```
# Loading our dataset
path <-"http://bit.ly/SupermarketDatasetII"
super <-read.transactions(path, sep = ",")
```

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

```
super
```

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

```
# Verifying the object's class
class(super)
```

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

```
# Previewing our first 5 transactions
inspect(super[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}
```

```
# Generating a summary of the dataset
summary(super)
```

```
## 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
```

The most commonly purchased item in our data is mineral water, eggs, spaghetti, french fries and chocolate.

We have 7,501 transactions containing 119 different items

There are 1,754 transactions where 1 item was bought and 1 transaction where 20 items were bought.

```
# Exploring the frequency of some articles
# and performing some operation in percentage terms of the total transactions
itemFrequency(super[,15:20],type = "absolute")
```

```
##     burgers      butter        cake  candy bars     carrots cauliflower
##         654         226         608          73         115          36
```

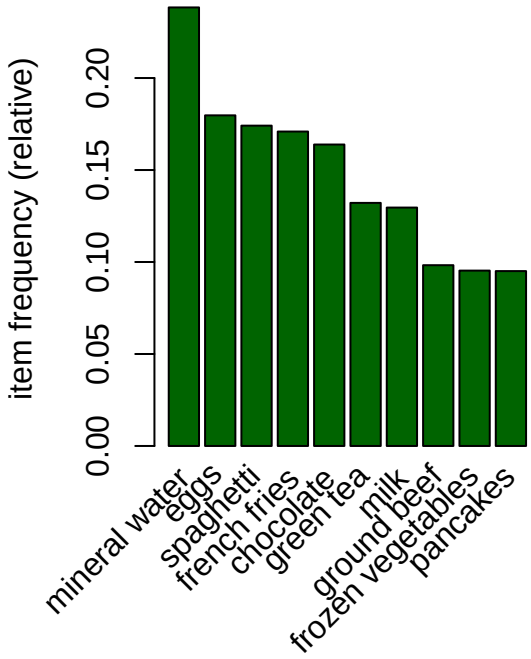
```
round(itemFrequency(super[, 15:20],type = "relative")*100,2)
```

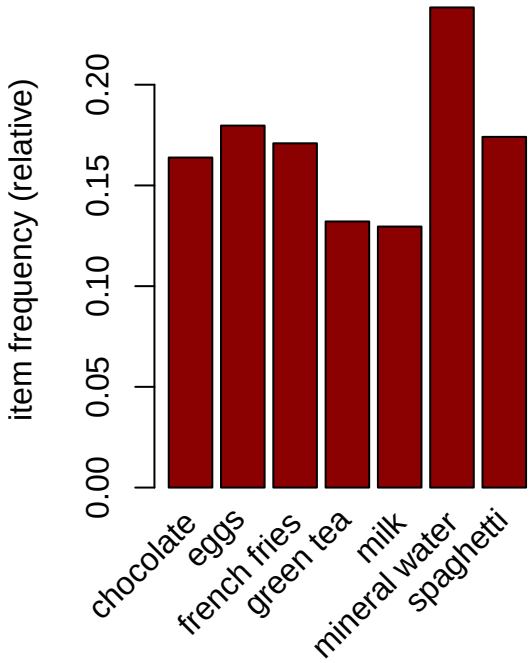
```
##     burgers      butter        cake  candy bars     carrots cauliflower
##        8.72        3.01        8.11        0.97        1.53        0.48
```

There are 654 burgers in our data which is 8.72% of the total number of transactions.

```
# Producing a chart of frequencies and filtering
# to consider only items with a minimum percentage of support
# Displaying top 10 most common items in the transactions dataset
# and the items whose relative importance is at least 10%
#
par(mfrow = c(1, 2))

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






```
# Let's see the rules we can get when we set minimum support at 0.10 and
# confidence at 80%
apriori (super, parameter = list(supp = 0.10, conf = 0.8))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5     0.1      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: 750
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.00s].
## sorting and recoding items ... [7 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].
```

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

We have no set of rules

```
# Let's see the rules we can get when we set confidence at support at 0.01
# confidence at 80%
apriori (super, parameter = list(supp = 0.01, conf = 0.8))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5    0.01      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: 75
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.01s].
## sorting and recoding items ... [75 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [0 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

## set of 0 rules
```

We have no rules

```
# Building a model based on association rules with minimum support at 0.001 and
# confidence at 80%
rules <- apriori (super, 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: 7
##
```

```
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.01s].
## sorting and recoding items ... [116 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 5 6 done [0.02s].
## writing ... [74 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules
```

```
## set of 74 rules
```

We have 74 rules

```
# Let's explore our model
summary(rules)
```

```
## set of 74 rules
##
## rule length distribution (lhs + rhs):sizes
##  3  4  5  6
## 15 42 16  1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   4.000   4.000   4.041   4.000   6.000
##
## summary of quality measures:
##     support           confidence        coverage             lift
##  Min.   :0.001067   Min.   :0.8000   Min.   :0.001067   Min.   : 3.356
##  1st Qu.:0.001067   1st Qu.:0.8000   1st Qu.:0.001333   1st Qu.: 3.432
##  Median :0.001133   Median :0.8333   Median :0.001333   Median : 3.795
##  Mean   :0.001256   Mean   :0.8504   Mean   :0.001479   Mean   : 4.823
##  3rd Qu.:0.001333   3rd Qu.:0.8889   3rd Qu.:0.001600   3rd Qu.: 4.877
##  Max.   :0.002533   Max.   :1.0000   Max.   :0.002666   Max.   :12.722
##      count
##  Min.   : 8.000
##  1st Qu.: 8.000
##  Median : 8.500
##  Mean   : 9.419
##  3rd Qu.:10.000
##  Max.   :19.000
##
## mining info:
##   data ntransactions support confidence
##  super          7501   0.001        0.8
##                                                                 call
##  apriori(data = super, parameter = list(supp = 0.001, conf = 0.8))
```

```
# Observe the first five rules
inspect(rules[1:5])
```

```
##     lhs                              rhs             support     confidence
```

```
## [1] {frozen smoothie, spinach}    => {mineral water} 0.001066524 0.8888889
## [2] {bacon, pancakes}             => {spaghetti}     0.001733102 0.8125000
## [3] {nonfat milk, turkey}         => {mineral water} 0.001199840 0.8181818
## [4] {ground beef, nonfat milk}    => {mineral water} 0.001599787 0.8571429
## [5] {mushroom cream sauce, pasta} => {escalope}      0.002532996 0.9500000
##     coverage    lift      count
## [1] 0.001199840  3.729058  8
## [2] 0.002133049  4.666587 13
## [3] 0.001466471  3.432428  9
## [4] 0.001866418  3.595877 12
## [5] 0.002666311 11.976387 19
```

Customers who buy frozen smoothie and spinach are nearly 89% likely to buy mineral water

```
# Let's sort our rules by level of confidence
rules<-sort(rules, by="confidence", decreasing=TRUE)
inspect(rules[1:5])
```

```
##     lhs                         rhs                support confidence    coverage      lift count
## [1] {french fries,
##      mushroom cream sauce,
##      pasta}                 => {escalope}      0.001066524       1.00 0.001066524 12.606723     8
## [2] {ground beef,
##      light cream,
##      olive oil}             => {mineral water} 0.001199840       1.00 0.001199840  4.195190     9
## [3] {cake,
##      meatballs,
##      mineral water}         => {milk}          0.001066524       1.00 0.001066524  7.717078     8
## [4] {cake,
##      olive oil,
##      shrimp}                => {mineral water} 0.001199840       1.00 0.001199840  4.195190     9
## [5] {mushroom cream sauce,
##      pasta}                 => {escalope}      0.002532996       0.95 0.002666311 11.976387    19
```

Customers who buy french fries,mushroom cream sauce and pasta are 100% likely to buy escalope

```
# Let's sort our rules by lift
inspect(sort(rules,by = "lift")[1:10])
```

```
##     lhs                         rhs                   support confidence    coverage      lift cou
## [1] {eggs,
##      mineral water,
##      pasta}                 => {shrimp}            0.001333156  0.9090909 0.001466471 12.722185
## [2] {french fries,
##      mushroom cream sauce,
##      pasta}                 => {escalope}          0.001066524  1.0000000 0.001066524 12.606723
## [3] {milk,
##      pasta}                 => {shrimp}            0.001599787  0.8571429 0.001866418 11.995203
## [4] {mushroom cream sauce,
##      pasta}                 => {escalope}          0.002532996  0.9500000 0.002666311 11.976387
## [5] {chocolate,
##      ground beef,
```

```
##       milk,
##       mineral water,
##       spaghetti}             => {frozen vegetables} 0.001066524  0.8888889 0.001199840  9.325253
## [6]  {herb & pepper,
##       mineral water,
##       rice}                  => {ground beef}       0.001333156  0.9090909 0.001466471  9.252498
## [7]  {grated cheese,
##       mineral water,
##       rice}                  => {ground beef}       0.001066524  0.8888889 0.001199840  9.046887
## [8]  {cake,
##       meatballs,
##       mineral water}         => {milk}              0.001066524  1.0000000 0.001066524  7.717078
## [9]  {escalope,
##       hot dogs,
##       mineral water}         => {milk}              0.001066524  0.8888889 0.001199840  6.859625
## [10] {meatballs,
##       whole wheat pasta}     => {milk}              0.001333156  0.8333333 0.001599787  6.430898
```

When a customer buys milk and pasta chances of buying shrimp increases by 11,00%

When a customer buys herb & pepper, mineral water and rice chances of buying ground beef increases by nearly 800%

```
# In an instance where we would be making a promotion relating to the sale of
# chocolate
# we could create a subset of rules concerning these products
# This would tell us the items that the customers bought before purchasing
# chocolate
sp <- subset(rules, subset = rhs %pin% "chocolate")

# Then order by confidence
sp<-sort(sp, by="confidence", decreasing=TRUE)
inspect(sp)
```

```
##     lhs                                  rhs         support     confidence
## [1] {escalope, french fries, shrimp} => {chocolate} 0.001066524 0.8888889
## [2] {red wine, tomato sauce}         => {chocolate} 0.001066524 0.8000000
##     coverage    lift     count
## [1] 0.001199840 5.425188 8
## [2] 0.001333156 4.882669 8
```

A purchase of red wine and tomato sauce results to a purchase of chocolate 80% of the time

```
# To answer the question customers who bought chocolate also bought ...
# We will put chocolate on lhs

# Subset the rules
sp.1 <- subset(rules, subset = lhs %pin% "chocolate")

# Order by confidence
sp.1 <-sort(sp.1, by="confidence", decreasing=TRUE)

# inspect top 5
inspect(sp.1)
```

```
##      lhs                     rhs                    support confidence    coverage     lift count
## [1]  {chocolate,
##       frozen vegetables,
##       olive oil,
##       shrimp}            => {mineral water}     0.001199840  0.9000000 0.001333156 3.775671     9
## [2]  {chocolate,
##       soup,
##       turkey}            => {mineral water}     0.001066524  0.8888889 0.001199840 3.729058     8
## [3]  {chocolate,
##       ground beef,
##       milk,
##       mineral water,
##       spaghetti}         => {frozen vegetables} 0.001066524  0.8888889 0.001199840 9.325253     8
## [4]  {chocolate,
##       frozen vegetables,
##       shrimp,
##       spaghetti}         => {mineral water}     0.001733102  0.8666667 0.001999733 3.635831    13
## [5]  {chocolate,
##       eggs,
##       frozen vegetables,
##       ground beef}       => {mineral water}     0.001466471  0.8461538 0.001733102 3.549776    11
## [6]  {chocolate,
##       eggs,
##       olive oil,
##       spaghetti}         => {mineral water}     0.001199840  0.8181818 0.001466471 3.432428     9
## [7]  {chocolate,
##       milk,
##       shrimp,
##       spaghetti}         => {mineral water}     0.001199840  0.8181818 0.001466471 3.432428     9
## [8]  {chocolate,
##       hot dogs,
##       milk}              => {mineral water}     0.001066524  0.8000000 0.001333156 3.356152     8
## [9]  {chocolate,
##       olive oil,
##       soup}              => {mineral water}     0.001599787  0.8000000 0.001999733 3.356152    12
## [10] {chocolate,
##       eggs,
##       milk,
##       olive oil}         => {mineral water}     0.001066524  0.8000000 0.001333156 3.356152     8
## [11] {chocolate,
##       french fries,
##       mineral water,
##       olive oil}         => {spaghetti}         0.001066524  0.8000000 0.001333156 4.594793     8
## [12] {chocolate,
##       frozen vegetables,
##       pancakes,
##       shrimp}            => {mineral water}     0.001066524  0.8000000 0.001333156 3.356152     8
```

a purchase of chocolate and a combination of other goods may result to a purchase of mineral water,frozen vegetables, spaghetti

```
# Determining items that the customers bought before purchasing
# chocolate
sp.3 <- subset(rules, subset = rhs %pin% "chocolate")
```

```
# Order by lift
sp.3<-sort(sp.3, by="lift", decreasing=TRUE)
inspect(sp.3)
```

```
##     lhs                                  rhs         support     confidence
## [1] {escalope, french fries, shrimp} => {chocolate} 0.001066524 0.8888889
## [2] {red wine, tomato sauce}         => {chocolate} 0.001066524 0.8000000
##     coverage    lift     count
## [1] 0.001199840 5.425188 8
## [2] 0.001333156 4.882669 8
```

There is a positive correlation between red wine, tomato sauce and chocolate and this is observed in 8 of our transactions

```
# Determine items that customers might buy who have previously bought
# chocolate

# Subset the rules
sp.4 <- subset(rules, subset = lhs %pin% "chocolate")

# Order by lift
sp.4 <-sort(sp.4, by="lift", decreasing=TRUE)

# inspect top 5
inspect(sp.4)
```

```
##      lhs                    rhs                     support confidence    coverage     lift count
## [1]  {chocolate,
##       ground beef,
##       milk,
##       mineral water,
##       spaghetti}          => {frozen vegetables} 0.001066524  0.8888889 0.001199840 9.325253     8
## [2]  {chocolate,
##       french fries,
##       mineral water,
##       olive oil}          => {spaghetti}         0.001066524  0.8000000 0.001333156 4.594793     8
## [3]  {chocolate,
##       frozen vegetables,
##       olive oil,
##       shrimp}             => {mineral water}     0.001199840  0.9000000 0.001333156 3.775671     9
## [4]  {chocolate,
##       soup,
##       turkey}             => {mineral water}     0.001066524  0.8888889 0.001199840 3.729058     8
## [5]  {chocolate,
##       frozen vegetables,
##       shrimp,
##       spaghetti}          => {mineral water}     0.001733102  0.8666667 0.001999733 3.635831    13
## [6]  {chocolate,
##       eggs,
##       frozen vegetables,
##       ground beef}        => {mineral water}     0.001466471  0.8461538 0.001733102 3.549776    11
## [7]  {chocolate,
```

```
##       eggs,
##       olive oil,
##       spaghetti}          => {mineral water}     0.001199840  0.8181818 0.001466471 3.432428     9
## [8]  {chocolate,
##       milk,
##       shrimp,
##       spaghetti}          => {mineral water}     0.001199840  0.8181818 0.001466471 3.432428     9
## [9]  {chocolate,
##       hot dogs,
##       milk}               => {mineral water}     0.001066524  0.8000000 0.001333156 3.356152     8
## [10] {chocolate,
##       olive oil,
##       soup}               => {mineral water}     0.001599787  0.8000000 0.001999733 3.356152    12
## [11] {chocolate,
##       eggs,
##       milk,
##       olive oil}          => {mineral water}     0.001066524  0.8000000 0.001333156 3.356152     8
## [12] {chocolate,
##       frozen vegetables,
##       pancakes,
##       shrimp}             => {mineral water}     0.001066524  0.8000000 0.001333156 3.356152     8
```

When a customer buys chocolate, olive oil and soup chances of buying mineral water increases by nearly 200%

People who bought chocolate and a combination of other goods are likely to buy spaghetti,frozen vegetables and mineral water.