Association Rules

IYLINE CHUMO

10/09/2021

Defining the Question

Performing analysis on Carrefour Kenya Dataset to help the marketing department on the most relevant marketing strategies that will result in the highest no. of sales (total price including tax).

We will create association rules that will allow us to identify relationships between variables in the dataset. The dataset comprises groups of items that will be associated with other then give insights thereafter.

#Loading the Dataset

Loading the arules package

```
suppressWarnings(
        suppressMessages(if
                         (!require(arules, quietly=TRUE))
                install.packages("arules")))
library(arules)
```

We will use read.transactions function 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/SupermarketDatasetII"

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

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

rules

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

previewing the top of our dataset

```
head(rules)

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

previewing the bottom of our dataset

```
tail(rules)

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

Data Exploration

```
str(rules)

## Formal class 'transactions' [package "arules"] with 3 slots
##   ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
##   .. .. ..@ i       : int [1:29358] 0 1 3 32 38 47 52 53 59 64 ...
##   .. .. ..@ p       : int [1:7502] 0 20 23 24 26 31 32 34 37 40 ...
##   .. .. ..@ Dim     : int [1:2] 119 7501
##   .. .. ..@ Dimnames:List of 2
##   .. .. .. ..$ : NULL
##   .. .. .. ..$ : NULL
##   .. .. ..@ factors : list()
##   ..@ itemInfo   :'data.frame':  119 obs. of  1 variable:
##   .. ..$ labels: chr [1:119] "almonds" "antioxydant juice" "asparagus"
"avocado" ...
##   ..@ itemsetInfo:'data.frame':  0 obs. of  0 variables

summary(rules)

## 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
```

Verifying the object’s class.

```
class(rules)

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

Previewing the first five transactions.

```
inspect(rules[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}
```

If we want to view the items that make up our dataset, we can convert it into a dataframe as shown below.

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

```
##                  Item
## 1             almonds
## 2  antioxydant juice
## 3           asparagus
## 4             avocado
## 5         babies food
## 6               bacon
## 7      barbecue sauce
## 8           black tea
## 9         blueberries
## 10         body spray
```

Generating a summary of our dataset. This will give us some information such as the most purchased items and the distribution of the item sets (no. of items purchased in each transaction), etc.

```
summary(rules)
```

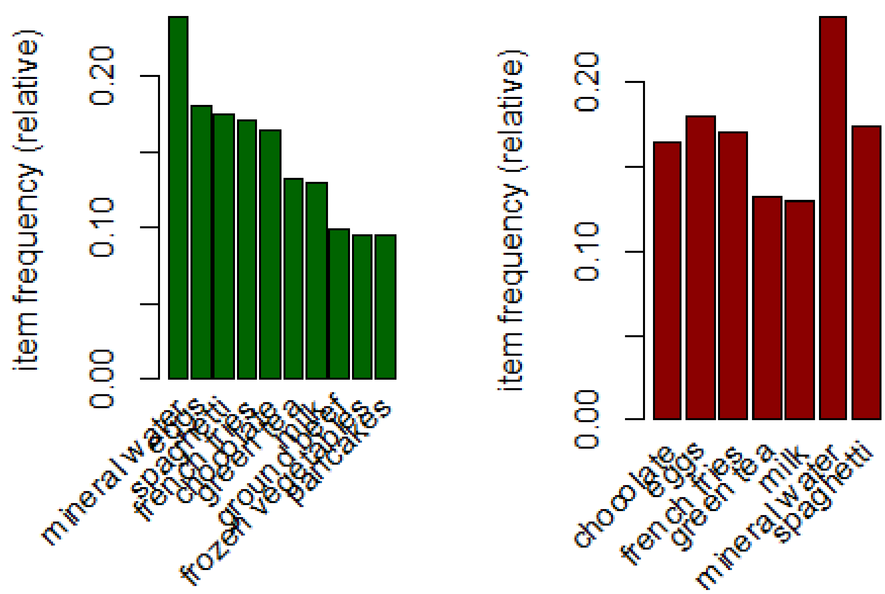
```
## transactions as itemMatrix in sparse format with
##  7501 rows (elements/itemsets/transactions) and
##  119 columns (items) and a density of 0.03288973 
## 
## most frequent items:
## mineral water          eggs     spaghetti  french fries     chocolate 
##          1788          1348          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
```

Exploring the frequency of some variables.

```
itemFrequency(rules[, 5:10],type = "absolute")

##     babies food           bacon barbecue sauce       black tea     blueberries
##              34              65             81             107              69
##      body spray
##              86

round(itemFrequency(rules[, 5:10],type = "relative")*100,2)

##     babies food           bacon barbecue sauce       black tea     blueberries
##            0.45            0.87           1.08            1.43            0.92
##      body spray
##            1.15
```

Producing a chart of frequencies and filtering to consider only items with a minimum percentage of support/ considering a top x of items. 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))
itemFrequencyPlot(rules, topN = 10,col="darkgreen")
itemFrequencyPlot(rules, support = 0.1,col="darkred")
```



From our plots, we see that mineral water is the most frequently purchased item, followed by eggs, spaghetti, french fries, chocolate, green tea, milk, ground beef, frozen vegetables and pancakes in that order.

Building a model based on association rules using the apriori function. We use Min Support as 0.001 and confidence as 0.8.

```
ass_rules <- apriori (rules, 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].

ass_rules

## set of 74 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 74 rules. However, in order to illustrate the sensitivity of the model to these two parameters, we will see what happens if we increase the support or lower the confidence.

Building a apriori model with Min Support as 0.002 and confidence as 0.8.

```
ass_rules2 <- apriori (rules,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: 15
```

```
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.00s].
## sorting and recoding items ... [115 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 done [0.01s].
## writing ... [2 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

ass_rules2

## set of 2 rules
```

Working with Min Support as 0.002 and confidence as 0.8, we get a set of 2 rules.

Building a apriori model with Min Support as 0.002 and confidence as 0.6.

```
ass_rules3 <- apriori (rules,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: 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 [0.01s].
## writing ... [545 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

ass_rules3

## set of 545 rules
```

This gives us a set of 43 rules.

In our first example, we increased the minimum support of 0.001 to 0.002 and model rules went from 74 to only 2. 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 74 to 545. 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 some exploration on our model through the use of the summary function. Upon running the code, the function would give us information about the model i.e. the size of rules, depending on the items that contain these rules. More statistical information such as support, lift and confidence is also provided.

```
summary(ass_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
##  rules          7501   0.001        0.8
```

Observing rules built in our model i.e the first 5 model rules.

```
inspect(ass_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
```

Interpretation of the first and fifth rule.

- If someone buys frozen smothie and spinach, they are 88% likely to buy mineral water too.

- If someone buys mushroom cream sauce and pasta, they are 95% likely to buy escalope too.

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

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

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

Conclusions

- From our plots, we see that mineral water is the most frequently purchased item, followed by eggs, spaghetti, french fries, chocolate, green tea, milk, ground beef, frozen vegetables and pancakes in that order.

- The marketing team should give discounts for the most purchased items so as to attract more customers.

- When arranging the lanes in the supermarket, goods which are mostly bought together should be arranged close together so as to make the shopping experience smooth for the customers.