Part3_Sales_Dataset: Association Analysis

Martha Irungu

9/18/2020

```
knitr::opts_chunk$set(error = TRUE)
```

Implementing association analysis

Part 3: Association Rules

In this section I will create association rules that will allow us to identify relationships between variables in the dataset. We have been provided with a dataset that comprises of groups of items that will be associated with others.We will provide insights for your analysis.

Association analysis is an unsupervised method that is used to discover patterns that occur within a given dataset by identifying relationships between observations and variables from a dataset. These relationships are defined by a set of rules that indicate groups of items that tend to be associated with others.

```
# We first we install the required arules library
#
Install.packages("arules")

## Error in Install.packages("arules"): could not find function "Install.pack
ages"
```

#Loading the arules library

```
library(arules)

## Loading required package: Matrix

##
## Attaching package: 'arules'

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

#Loading and previewing the dataset

```
transactions<-read.transactions("~/Desktop/Sales_Dataset_2.csv", sep = ",")

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

transactions
```

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

#We observe that we have 7501 transactions(rows) and 119 items/products(columns)

#Structure of the dataset

```r
str(transactions)
```

```
## 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" "av
ocado" ...
##   ..@ itemsetInfo:'data.frame':  0 obs. of  0 variables
```

#Details are as tabulated

```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"
```

#We observe transactions and attribute package as arules

```r
# Previewing our 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}
```

#The first 5 transactions are as tabulated. The 1st transaction has a list of many products that were purchased together. The second transaction has burgers,eggs and meatballs, 3rd has chutney only, fourth has avocado and turkey.

#Previewing the 10 items in our dataset in a different way

```
items<-as.data.frame(itemLabels(transactions))
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
```

#We observe almonds, antioxydant juice, asparagus, avocado and babies food are the first 5 items

```
# 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
##  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
```

#We observe that mineral water is the top 10 most frequently purchased items, with 1,788 frequency followed by eggs with 1,348 and spagheti with 1,306.

```r
# Exploring the frequency of some products
# 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")
```

```
##   black tea blueberries  body spray
##         107          69          86
```

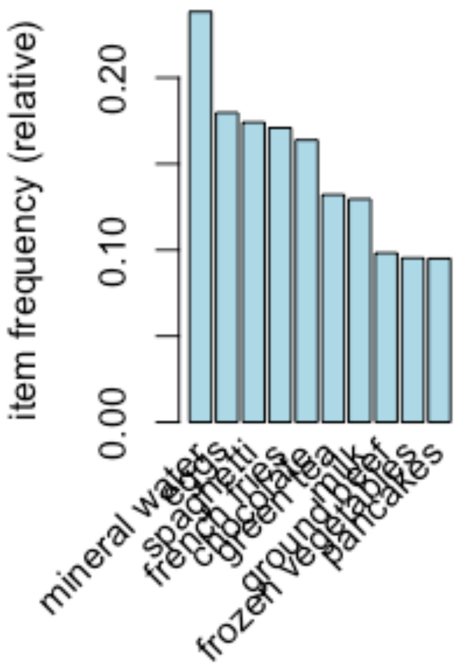
```r
round(itemFrequency(transactions[, 8:10],type = "relative")*100,2)
```

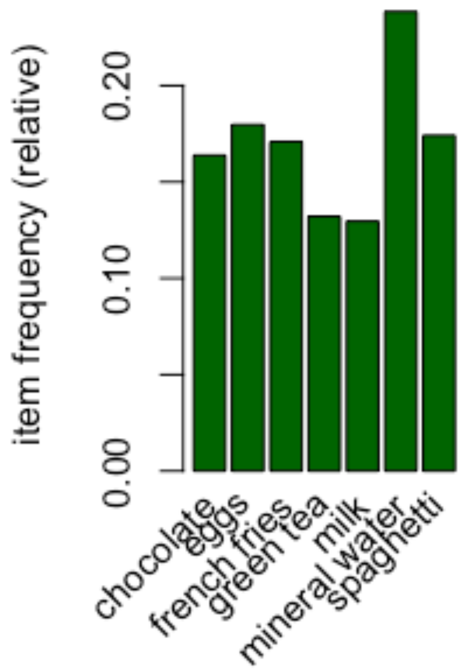
```
##   black tea blueberries  body spray
##        1.43        0.92        1.15
```

#we observe that black tea was purchased 107 times contributing to 1.43%

```r
# Producing a chart of frequencies and fitering
# to consider only items with a minimum percentage
# of support/ considering a top 10 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))

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





#We observe the top 10 items on the left graph, we have mineral water, eggs and spaghetti as top 3 and items with at least support of 10% on the right hand side graph

#Building the model

```
# 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: 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 [0.01s].
## writing ... [74 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

rules

## set of 74 rules
```

#We observe that we have 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 Minimum support
# and confidence as 0.8 we obtained 74 rules.
# However, in order to illustrate the sensitivity of the model to these two p
arameters,
# we will see what happens if we increase the support or lower the confidence
level
#

# 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: 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].
```

```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: 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 [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.01s].
## writing ... [545 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```r
rules2
```

```
## set of 2 rules
```

```r
rules3
```

```
## set of 545 rules
```

#In first model, we increased the minimum support of 0.001 to 0.002 and model rules went from 74 rules to only 2 rules. This would lead us to understand that using a high level of support can make the model lose interesting rules. #In the second model, 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.

#Explore our model by running the summary function

```
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
##  transactions          7501   0.001        0.8
```

#This gives us summary of the model as tabulated, statistical information such as support, lift and confidence is also provided.

#Observing rules built in our model. i.e first 5 model 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
```

#We observe that:

#Rule 1: If someone buys frozen smoothies and spinach, they are 88.8% likely to buy mineral water too

#Rule 2: If someone buys bacon and pancakes, they are 81% likely to buy spaghetti too

#Ordering the rules

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

```
##     lhs                                           rhs             support
## [1] {french fries,mushroom cream sauce,pasta} => {escalope}      0.0010665
24
## [2] {ground beef,light cream,olive oil}       => {mineral water} 0.0011998
40
## [3] {cake,meatballs,mineral water}            => {milk}          0.0010665
24
## [4] {cake,olive oil,shrimp}                   => {mineral water} 0.0011998
40
## [5] {mushroom cream sauce,pasta}              => {escalope}      0.0025329
96
##     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
```

#The given five rules have a confidence of 100 for the first 4v and the 5th has 95% confidence

#Checking promotion on milk

```
# We create a subset of rules concerning these products
# This would tell us the items that the customers bought before purchasing mi
lk
```

```r
milk<- subset(rules, subset = rhs %pin% "milk")

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

```
##     lhs                                  rhs    support     confidence
## [1] {cake,meatballs,mineral water}    => {milk} 0.001066524 1.0000000
## [2] {escalope,hot dogs,mineral water} => {milk} 0.001066524 0.8888889
## [3] {meatballs,whole wheat pasta}     => {milk} 0.001333156 0.8333333
## [4] {black tea,frozen smoothie}       => {milk} 0.001199840 0.8181818
## [5] {burgers,ground beef,olive oil}   => {milk} 0.001066524 0.8000000
##     coverage    lift     count
## [1] 0.001066524 7.717078  8
## [2] 0.001199840 6.859625  8
## [3] 0.001599787 6.430898 10
## [4] 0.001466471 6.313973  9
## [5] 0.001333156 6.173663  8
```

#We observe that in rule 1 a customer would buy rice and sugar, they are 100% likely to buy whole milk #In rule 2 if a customer buys canned fish and hygiene articles they are 100% likley to buy whole milk and so on.

#Determine items that customers might buy who have previously bought milk?

```r
# Subset the rules
milk <- subset(rules, subset = lhs %pin% "milk")

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

# inspect top 5
inspect(milk[1:5])
```

```
##     lhs                     rhs                        support  confidence    c
overage     lift count
## [1] {frozen vegetables,
##      milk,
##      spaghetti,
##      turkey}             => {mineral water}     0.001199840  0.9000000 0.00
1333156 3.775671     9
## [2] {cake,
##      meatballs,
##      milk}               => {mineral water}     0.001066524  0.8888889 0.00
1199840 3.729058     8
## [3] {burgers,
##      milk,
##      salmon}             => {spaghetti}         0.001066524  0.8888889 0.00
```

```
1199840 5.105326     8
## [4] {chocolate,
##      ground beef,
##      milk,
##      mineral water,
##      spaghetti}         => {frozen vegetables} 0.001066524  0.8888889 0.00
1199840 9.325253     8
## [5] {ground beef,
##      nonfat milk}       => {mineral water}     0.001599787  0.8571429 0.00
1866418 3.595877    12
```

#In rule 1 we see a customer who buys grapes,tropical fruit, yoghurt and whole milk is 100% likely to purchase other vegetables.

Conclusion

#Mineral water was a popular product and most customers who bought other items were likely to purchase mineral water as well.This item can be placed together with associated items to increase sales.

#The supermarket can also accelerate sales of other products e.g whole milk by grouping it with other products such as rice ans sugar, canned fish, hygiene articles among others that showed evidence of if purchased, the customers are likely to purchase whole milk as well.