Association Rules in R

Dennis Kiarie

9/11/2021

```
library(tinytex)
```

1. Define the Question

1.1 Research Question

Our Research seeks to create association rules that will allow us to identify relationships between variables in the dataset.

1.2 Metric of Success

To perform analysis and provide insights on how dataset variables are related to each other.

1.3 The Context

You are a Data analyst at Carrefour Kenya and are currently undertaking a project that will inform the marketing department on the most relevant marketing strategies that will result in the highest no. of sales (total price including tax). Your project has been divided into four parts where you'll explore a recent marketing dataset by performing various unsupervised learning techniques and later providing recommendations based on your insights.

1.4 Experimental Design

1. Loading Data into RStudio.
2. Checking the Data.
3. Tidying the Data.
4. Conducting Exploratory Data Analysis i.e Univariate, Bivariate and Multivariate Analysis.
5. Association Rules.
6. Implement the Solution
7. Challenge the Solution
8. Follow up Questions

1.5 Data Relevance

The data provided its appropriate for our analysis. The dataset for this analysis can be found in this link:[http://bit.ly/SupermarketDatasetII]

Description

The dataset consists of 1000 records and 6 features.

2.Data Preparation

```
## Importing libraries
#---
#
```

```
library(pacman)
library(data.table)
pacman :: p_load(pacman,ggbiplot,plyr, dplyr,scales, readr, grid,factoextra, GGally,DataExplorer, ggplo
```

```
## Installing package into 'C:/Users/Denoo/OneDrive/Documents/R/win-library/4.1'
## (as 'lib' is unspecified)
```

```
## Warning: package 'FSelecto' is not available for this version of R
##
## A version of this package for your version of R might be available elsewhere,
## see the ideas at
## https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages
```

```
## Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contri
##   cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/4.1/PACKAGES'
```

```
## Warning: 'BiocManager' not available.  Could not check Bioconductor.
##
## Please use `install.packages('BiocManager')` and then retry.
```

```
## Warning in p_install(package, character.only = TRUE, ...):
```

```
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'FSelecto'
```

```
## Warning in pacman::p_load(pacman, ggbiplot, plyr, dplyr, scales, readr, : Failed to install/load:
## FSelecto
```

```
theme_set(theme_classic())
options(warn = -1)
```

```
path <-"http://bit.ly/SupermarketDatasetII"
df<-read.transactions(path, sep = ",")
df
```

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

3. Checking the data

```
##we check for the shape of the data
#---
#
dim(df)
```

```
## [1] 7501  119
```

```
#our dataset for analysis has 7501 records and 119 columns
```

```
## we check for the number of rows and columns
#---
#
cat("Rows:", nrow(df), "\nCols:", ncol(df))
```

```
## Rows: 7501
## Cols: 119
```

```
##we check if datatypes are appropriate
#---
#
glimpse(df)
```

```
## Formal class 'transactions' [package "arules"] with 3 slots
##   ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
##   ..@ 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
```

```
##we check for the number of columns
#---
#
length(df)
```

```
## [1] 7501
```

```
##we check the column names for easier reference
#---
#
colnames(df)
```

```
##   [1] "almonds"              "antioxydant juice"    "asparagus"
##   [4] "avocado"              "babies food"          "bacon"
##   [7] "barbecue sauce"       "black tea"            "blueberries"
##  [10] "body spray"           "bramble"              "brownies"
##  [13] "bug spray"            "burger sauce"         "burgers"
##  [16] "butter"               "cake"                 "candy bars"
##  [19] "carrots"              "cauliflower"          "cereals"
##  [22] "champagne"            "chicken"              "chili"
##  [25] "chocolate"            "chocolate bread"      "chutney"
##  [28] "cider"                "clothes accessories"  "cookies"
##  [31] "cooking oil"          "corn"                 "cottage cheese"
##  [34] "cream"                "dessert wine"         "eggplant"
##  [37] "eggs"                 "energy bar"           "energy drink"
##  [40] "escalope"             "extra dark chocolate" "flax seed"
##  [43] "french fries"         "french wine"          "fresh bread"
##  [46] "fresh tuna"           "fromage blanc"        "frozen smoothie"
##  [49] "frozen vegetables"    "gluten free bar"      "grated cheese"
##  [52] "green beans"          "green grapes"         "green tea"
```

```
##  [55] "ground beef"          "gums"                "ham"
##  [58] "hand protein bar"     "herb & pepper"       "honey"
##  [61] "hot dogs"             "ketchup"             "light cream"
##  [64] "light mayo"           "low fat yogurt"      "magazines"
##  [67] "mashed potato"        "mayonnaise"          "meatballs"
##  [70] "melons"               "milk"                "mineral water"
##  [73] "mint"                 "mint green tea"      "muffins"
##  [76] "mushroom cream sauce" "napkins"             "nonfat milk"
##  [79] "oatmeal"              "oil"                 "olive oil"
##  [82] "pancakes"             "parmesan cheese"     "pasta"
##  [85] "pepper"               "pet food"            "pickles"
##  [88] "protein bar"          "red wine"            "rice"
##  [91] "salad"                "salmon"              "salt"
##  [94] "sandwich"             "shallot"             "shampoo"
##  [97] "shrimp"               "soda"                "soup"
## [100] "spaghetti"            "sparkling water"     "spinach"
## [103] "strawberries"         "strong cheese"       "tea"
## [106] "tomato juice"         "tomato sauce"        "tomatoes"
## [109] "toothpaste"           "turkey"              "vegetables mix"
## [112] "water spray"          "white wine"          "whole weat flour"
## [115] "whole wheat pasta"    "whole wheat rice"    "yams"
## [118] "yogurt cake"          "zucchini"
```

```
##we verify the class of the data
class(df)
```

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

```
## we Preview our first 5 transactions
#---
#
inspect(df[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}
```

```
##we preview the items of the dataset
#---
#
items<-as.data.frame(itemLabels(df))
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 explore the frequency of some articles
#---
#
itemFrequency(df[, 8:10],type = "absolute")
```

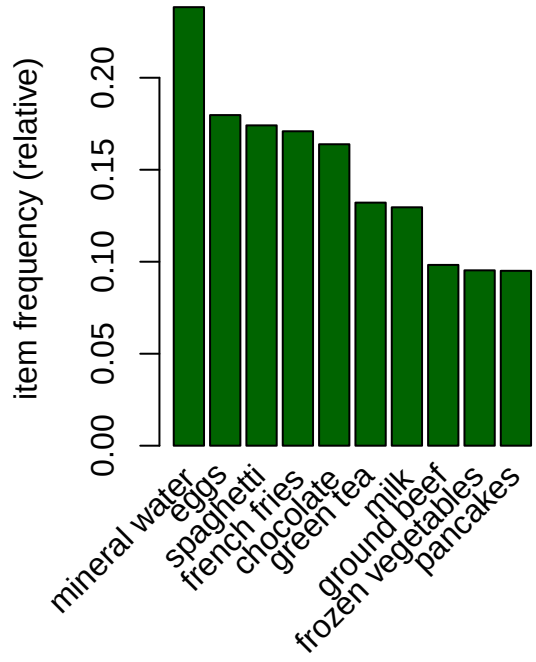
```
##   black tea blueberries  body spray
##         107          69          86
```

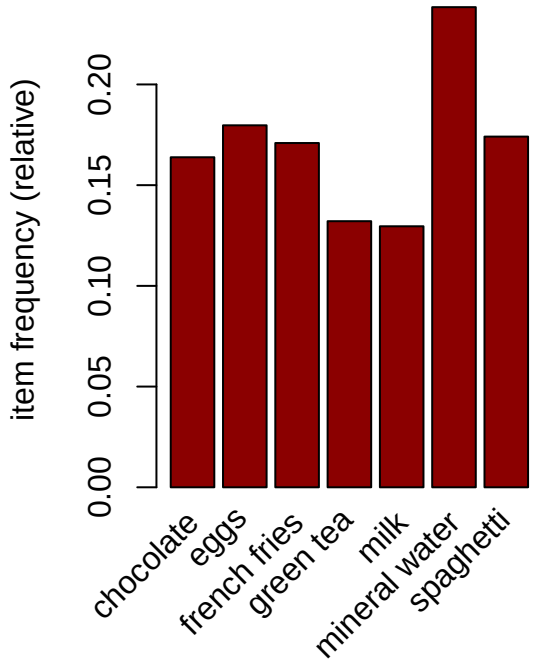
```
round(itemFrequency(df[, 8:10],type = "relative")*100,2)
```

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

```
##we Produce a chart of frequencies and filtering
# 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))
```

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






```
## we Build a model based on association rules using the apriori function
# We use Min Support as 0.001 and confidence as 0.8
#---
#
rules <- apriori (df, 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
```

```
## using the  measures of significance and interest on the rules,determining which ones are interesting
# Building a apriori model with Min Support as 0.002 and confidence as 0.8.
#---
rules2 <- apriori (df,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 [0.00s].
## writing ... [2 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules2
```

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

```
# Building apriori model with Min Support as 0.002 and confidence as 0.6.
rules3 <- apriori (df, 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 [0.00s].
## 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 [0.02s].
## writing ... [545 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules3
```

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

Conclusion

The first model had 74 rules while the second had 2 rules with a confidence level of 0.8 but different minimum supports. The same applies to the third that had 545 rules. Therefore, we conclude that a higher support level equals a loss of intresting rules while a low confidence level equals a higher number of rules, though not all of them will be useful.

```
## performing an exploration of our model through the use of 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
##    df          7501   0.001        0.8
```

```
##we Observe 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
```

conclusion

Interpretation of the rules: 1.If someone buys frozen smoothie and spinach, they are 89% likely to buy mineral water too. 2.If someone buys bacon and pancakes, they are 81% likely to buy spaghetti too. 3.If someone buys nonfat milk and turkey, they are 82% likely to buy mineral water too. 4.If someone buys ground beef and nonfat milk, they are 86% likely to buy mineral water too. 5.If someone buys frozen 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.
rules<-sort(rules, by="confidence", decreasing=TRUE)
inspect(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
```

Conclusion

Four of the given five rules have a confidence of 100 and the fifth rule has a confidence of 95

```
## If we're interested in making a promotion relating to the sale of mineral_water,
# we could create a subset of rules concerning these products
# ---
# This would tell us the items that the customers bought before purchasing mineral_water
# ---
#
mineral_water<- subset(rules, subset = rhs %pin% "mineral water")

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

```
## set of 41 rules
```

```
inspect(mineral_water[1:5])
```

```
##     lhs                    rhs                 support confidence    coverage     lift count
## [1] {ground beef,
##      light cream,
##      olive oil}         => {mineral water} 0.001199840  1.0000000 0.001199840 4.195190     9
## [2] {cake,
##      olive oil,
##      shrimp}            => {mineral water} 0.001199840  1.0000000 0.001199840 4.195190     9
## [3] {red wine,
##      soup}              => {mineral water} 0.001866418  0.9333333 0.001999733 3.915511    14
## [4] {ground beef,
##      pancakes,
##      whole wheat rice}  => {mineral water} 0.001333156  0.9090909 0.001466471 3.813809    10
## [5] {frozen vegetables,
##      milk,
##      spaghetti,
##      turkey}            => {mineral water} 0.001199840  0.9000000 0.001333156 3.775671     9
```

```
##What if we wanted to determine items that customers might buy
# who have previously bought mineral_water?
# ---
#
# Subset the rules
mineral_water <- subset(rules, subset = lhs %pin% "mineral_water")

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

# inspect top 5
inspect(mineral_water)
```