

# Association Analysis

Brian Onchweri

2022-06-10

## Part 2: Association Rules

This section will require that we create association rules that will allow us to identify relationships between variables in the dataset. We are provided with a separate dataset that comprises groups of items that will be associated with others. Just like in the other sections, we will also be required to provide insights for our analysis.

### Loading Required Libraries

```
library(relaimpo)

## Loading required package: MASS

## Loading required package: boot

## Loading required package: survey

## Loading required package: grid

## Loading required package: Matrix

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:boot':
##
##      aml

##
## Attaching package: 'survey'

## The following object is masked from 'package:graphics':
##
##      dotchart
```

```

## Loading required package: mitools

## This is the global version of package relaimpo.

## If you are a non-US user, a version with the interesting additional metric pmvd is available
## from Ulrike Groempings web site at prof.beuth-hochschule.de/groemping.

library(data.table)
library(ggplot2) # Data visualization
library(ggthemes) # Plot themes
library(plotly) # Interactive data visualizations

##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##   last_plot

## The following object is masked from 'package:MASS':
##
##   select

## The following object is masked from 'package:stats':
##
##   filter

## The following object is masked from 'package:graphics':
##
##   layout

library(dplyr) # Data manipulation

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##   between, first, last

## The following object is masked from 'package:MASS':
##
##   select

## The following objects are masked from 'package:stats':
##
##   filter, lag

## The following objects are masked from 'package:base':
##
##   intersect, setdiff, setequal, union

```

```
library(psych) # Will be used for correlation visualization
```

```
##  
## Attaching package: 'psych'  
  
## The following objects are masked from 'package:ggplot2':  
##  
##    %+%, alpha  
  
## The following object is masked from 'package:boot':  
##  
##    logit
```

```
library(arules)# for association
```

```
##  
## Attaching package: 'arules'  
  
## The following object is masked from 'package:dplyr':  
##  
##    recode  
  
## The following objects are masked from 'package:base':  
##  
##    abbreviate, write
```

## Reading the URL

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

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

## Displaying first few rows of our Dataset

```
head(df)
```

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

```
# previewing the column names  
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"	

```
dim(df)
```

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

```
# Previewing 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}
```

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

```
## 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 articles
itemFrequency(df[, 8:10],type = "absolute")
```

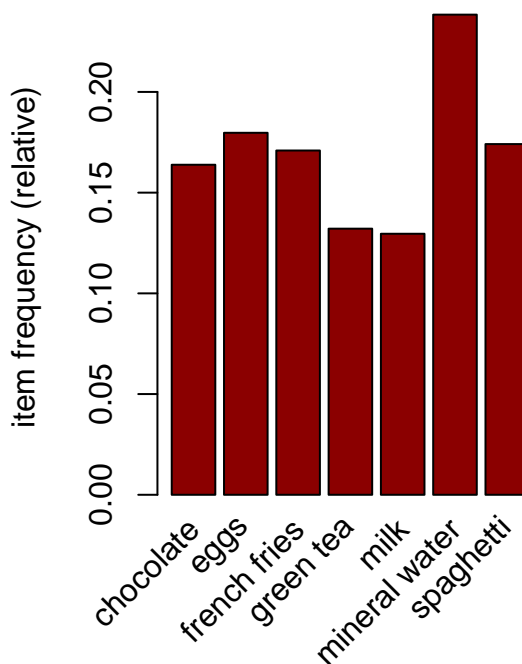
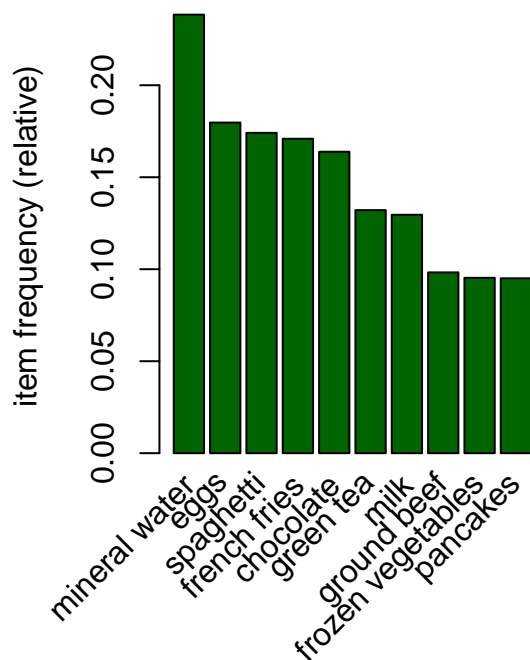
```
##    black tea blueberries  body spray
##         107          69          86
```

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

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

```
# Producing 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))
```

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



```
# 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 (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 [0.04s].
```

```
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].
```

```
# 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 [0.01s].
## writing ... [545 rule(s)] done [0.00s].
## creating S4 object ... done [0.00s].
```

```
rules2
```

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

```
rules3
```

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

```
# performing an exploration of our model through the use of the summary function as shown
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
## call
## apriori(data = df, parameter = list(supp = 0.001, conf = 0.8))
```

```
# 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
```

```
# 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 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
```

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

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

```
## set of 5 rules
```

```
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
```

```
# Determining items that customers might buy if they bought milk.
```

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

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

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

```
##      lhs                                rhs      support
## [1] {chocolate, hot dogs, milk}      => {mineral water} 0.001066524
## [2] {avocado, burgers, milk}         => {spaghetti}     0.001066524
## [3] {cookies, green tea, milk}       => {french fries}  0.001066524
## [4] {cake, eggs, milk, turkey}       => {mineral water} 0.001066524
## [5] {chocolate, eggs, milk, olive oil} => {mineral water} 0.001066524
##      confidence coverage      lift      count
## [1] 0.8          0.001333156 3.356152      8
```

```
## [2] 0.8      0.001333156 4.594793 8
## [3] 0.8      0.001333156 4.680811 8
## [4] 0.8      0.001333156 3.356152 8
## [5] 0.8      0.001333156 3.356152 8
```

###Conclusion

Interpretation of the rules:

1: If someone buys frozen smoothie and spinach, they are 89% likely to buy mineral water too 1: If someone buys bacon and pancakes, they are 81% likely to buy spaghetti too 1: If someone buys nonfat milk and turkey, they are 82% likely to buy mineral water too 1: If someone buys ground beef and nonfat milk, they are 86% likely to buy mineral water too 1: If someone buys frozen mushroom cream sauce and pasta, they are 95% likely to buy escalope too