title: “Part 3 - Association Rules” author: “Joy Muli” date: “11/13/2020” output: md\_document: variant: markdown\_github

# Loading required libraries   
library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2 v purrr 0.3.4  
## v tibble 3.0.4 v dplyr 1.0.2  
## v tidyr 1.1.2 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(arules)

## Loading required package: Matrix

##   
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':  
##   
## expand, pack, unpack

##   
## Attaching package: 'arules'

## The following object is masked from 'package:dplyr':  
##   
## recode

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

# Reading the data   
#df3 <- read.csv(file = 'http://bit.ly/SupermarketDatasetII')  
path<- "http://bit.ly/SupermarketDatasetII"  
sales <- read.transactions(path, sep= ",")

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

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

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

sales

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

# Checking out the structure  
str(sales)

## 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

# Checking the summary  
summary(sales)

## 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

# Previewing the items that make up our dataset,  
items<-as.data.frame(itemLabels(sales))  
colnames(items) <- "Item"  
head(items, 10)

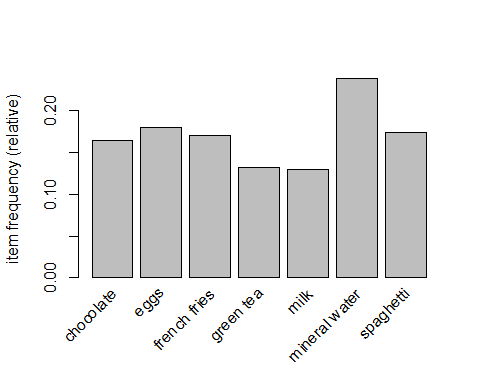
## Item  
## 1 almonds  
## 2 antioxydant juice  
## 3 asparagus  
## 4 avocado  
## 5 babies food  
## 6 bacon  
## 7 barbecue sauce  
## 8 black tea  
## 9 blueberries  
## 10 body spray

# Examining the frequency of certain items

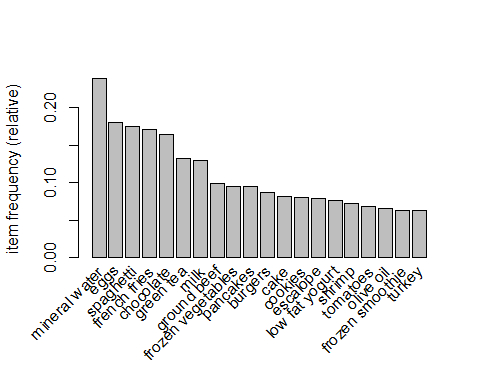
itemFrequency(sales[, 1:3])

## almonds antioxydant juice asparagus   
## 0.020397280 0.008932142 0.004799360

# Plotting the frequency of items  
itemFrequencyPlot(sales, support = 0.1)

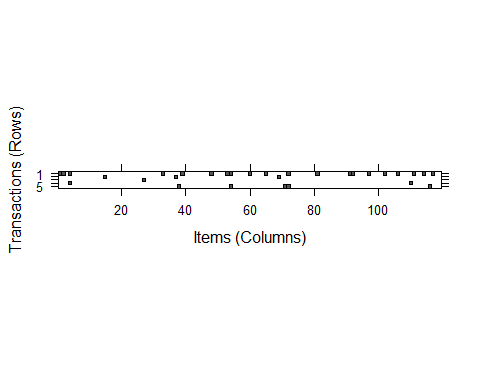


itemFrequencyPlot(sales, topN = 20)



# A visualization for the sparse matrix for the first 5 items

image(sales[1:5])



# Training a model on the data

apriori(sales)

## 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 [0.02s].  
## sorting and recoding items ... [7 item(s)] done [0.00s].  
## creating transaction tree ... done [0.01s].  
## 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

# Default settings results in zero rules learned

# Setting up better cofidence levels to learn more rules

salerules <- apriori(sales, parameter = list(support =  
 0.001, confidence = 0.8, minlen = 2))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.8 0.1 1 none FALSE TRUE 5 0.001 2  
## 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 [0.02s].  
## 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 [0.04s].  
## writing ... [74 rule(s)] done [0.00s].  
## creating S4 object ... done [0.01s].

# 74 rules

# we will see what happens if we increase the support or lower the confidence level

# Increasing the confidence level  
rules2 <- apriori (sales,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 [0.01s].  
## sorting and recoding items ... [115 item(s)] done [0.00s].  
## creating transaction tree ... done [0.01s].  
## checking subsets of size 1 2 3 4 5 done [0.02s].  
## writing ... [2 rule(s)] done [0.00s].  
## creating S4 object ... done [0.01s].

# 2 rules

# Decreasing the confidence level  
rules3 <- apriori (sales, 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 [0.02s].  
## 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].

# 545 rules

salerules

## set of 74 rules

rules2

## set of 2 rules

rules3

## set of 545 rules

In our second rule, 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 third rule, we decreased the minimum confidence level to 0.6 and the number of model rules went from 2 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.

# Evaluating model performance

# summary of grocery association rules  
summary(salerules)

## 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  
## sales 7501 0.001 0.8

# Looking at the first 5 rules

inspect(salerules[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

The first rule can be read in plain language as, “if a customer buys frozen smoothie,spinach, they will also buy mineral water.” With support of 0.0011 and confidence of 0.89, we can determine that this rule covers 0.1 percent of the transactions and is correct in 88 percent of purchases involving frozen smoothie,spinach. The lift value tells us how much more likely a customer is to buy mineral water relative to the average customer, given that he or she bought a frozen smoothie,spinach.

# Improving model performance

# Sorting salerules by lift  
inspect(sort(salerules, by = "lift")[1:5])

## lhs rhs support confidence coverage lift count  
## [1] {eggs,   
## mineral water,   
## pasta} => {shrimp} 0.001333156 0.9090909 0.001466471 12.722185 10  
## [2] {french fries,   
## mushroom cream sauce,   
## pasta} => {escalope} 0.001066524 1.0000000 0.001066524 12.606723 8  
## [3] {milk,   
## pasta} => {shrimp} 0.001599787 0.8571429 0.001866418 11.995203 12  
## [4] {mushroom cream sauce,   
## pasta} => {escalope} 0.002532996 0.9500000 0.002666311 11.976387 19  
## [5] {chocolate,   
## ground beef,   
## milk,   
## mineral water,   
## spaghetti} => {frozen vegetables} 0.001066524 0.8888889 0.001199840 9.325253 8

The first rule in these new rules suggest that a person who buys eggs, mineral water and pasta is 12 times more likely to buy shrimp than an average customer. Also the second rule suggest that escalope is 12 times more likely to be bought with french fries, mushroom cream sauce and pasta.

# Finding subset of rules that contain pasta

pastarules <- subset(salerules, items %in% "pasta")  
inspect(pastarules)

## lhs rhs support   
## [1] {mushroom cream sauce,pasta} => {escalope} 0.002532996  
## [2] {milk,pasta} => {shrimp} 0.001599787  
## [3] {french fries,mushroom cream sauce,pasta} => {escalope} 0.001066524  
## [4] {mineral water,pasta,shrimp} => {eggs} 0.001333156  
## [5] {eggs,mineral water,pasta} => {shrimp} 0.001333156  
## confidence coverage lift count  
## [1] 0.9500000 0.002666311 11.976387 19   
## [2] 0.8571429 0.001866418 11.995203 12   
## [3] 1.0000000 0.001066524 12.606723 8   
## [4] 0.8333333 0.001599787 4.637117 10   
## [5] 0.9090909 0.001466471 12.722185 10

A basket with mushroom cream sauce and pasta is modt likely to have escalope (rule1), milk & pasta are most likely to be bought with shrimp (rule2)

water <- subset(salerules, items %in% "mineral water")  
inspect(water)

## lhs rhs support confidence coverage lift count  
## [1] {frozen smoothie,   
## spinach} => {mineral water} 0.001066524 0.8888889 0.001199840 3.729058 8  
## [2] {nonfat milk,   
## turkey} => {mineral water} 0.001199840 0.8181818 0.001466471 3.432428 9  
## [3] {ground beef,   
## nonfat milk} => {mineral water} 0.001599787 0.8571429 0.001866418 3.595877 12  
## [4] {cooking oil,   
## fromage blanc} => {mineral water} 0.001199840 0.8181818 0.001466471 3.432428 9  
## [5] {black tea,   
## salmon} => {mineral water} 0.001066524 0.8000000 0.001333156 3.356152 8  
## [6] {pancakes,   
## tomato sauce} => {mineral water} 0.001066524 0.8000000 0.001333156 3.356152 8  
## [7] {red wine,   
## soup} => {mineral water} 0.001866418 0.9333333 0.001999733 3.915511 14  
## [8] {turkey,   
## whole wheat pasta} => {mineral water} 0.001466471 0.8461538 0.001733102 3.549776 11  
## [9] {milk,   
## spaghetti,   
## strong cheese} => {mineral water} 0.001066524 0.8000000 0.001333156 3.356152 8  
## [10] {mineral water,   
## pasta,   
## shrimp} => {eggs} 0.001333156 0.8333333 0.001599787 4.637117 10  
## [11] {eggs,   
## mineral water,   
## pasta} => {shrimp} 0.001333156 0.9090909 0.001466471 12.722185 10  
## [12] {ground beef,   
## light cream,   
## olive oil} => {mineral water} 0.001199840 1.0000000 0.001199840 4.195190 9  
## [13] {light cream,   
## mineral water,   
## shrimp} => {spaghetti} 0.001066524 0.8888889 0.001199840 5.105326 8  
## [14] {cake,   
## meatballs,   
## milk} => {mineral water} 0.001066524 0.8888889 0.001199840 3.729058 8  
## [15] {cake,   
## meatballs,   
## mineral water} => {milk} 0.001066524 1.0000000 0.001066524 7.717078 8  
## [16] {herb & pepper,   
## mineral water,   
## rice} => {ground beef} 0.001333156 0.9090909 0.001466471 9.252498 10  
## [17] {grated cheese,   
## ground beef,   
## rice} => {mineral water} 0.001066524 0.8000000 0.001333156 3.356152 8  
## [18] {grated cheese,   
## mineral water,   
## rice} => {ground beef} 0.001066524 0.8888889 0.001199840 9.046887 8  
## [19] {milk,   
## mineral water,   
## parmesan cheese} => {spaghetti} 0.001066524 0.8000000 0.001333156 4.594793 8  
## [20] {oil,   
## shrimp,   
## spaghetti} => {mineral water} 0.001066524 0.8000000 0.001333156 3.356152 8  
## [21] {cooking oil,   
## mineral water,   
## red wine} => {spaghetti} 0.001066524 0.8000000 0.001333156 4.594793 8  
## [22] {cake,   
## olive oil,   
## whole wheat pasta} => {mineral water} 0.001066524 0.8888889 0.001199840 3.729058 8  
## [23] {escalope,   
## hot dogs,   
## milk} => {mineral water} 0.001066524 0.8000000 0.001333156 3.356152 8  
## [24] {escalope,   
## hot dogs,   
## mineral water} => {milk} 0.001066524 0.8888889 0.001199840 6.859625 8  
## [25] {chocolate,   
## hot dogs,   
## milk} => {mineral water} 0.001066524 0.8000000 0.001333156 3.356152 8  
## [26] {brownies,   
## eggs,   
## ground beef} => {mineral water} 0.001066524 0.8888889 0.001199840 3.729058 8  
## [27] {chicken,   
## fresh bread,   
## pancakes} => {mineral water} 0.001066524 0.8888889 0.001199840 3.729058 8  
## [28] {french fries,   
## herb & pepper,   
## milk} => {mineral water} 0.001199840 0.8181818 0.001466471 3.432428 9  
## [29] {chocolate,   
## soup,   
## turkey} => {mineral water} 0.001066524 0.8888889 0.001199840 3.729058 8  
## [30] {olive oil,   
## soup,   
## tomatoes} => {mineral water} 0.001333156 0.8333333 0.001599787 3.495992 10  
## [31] {frozen vegetables,   
## olive oil,   
## soup} => {mineral water} 0.001733102 0.8125000 0.002133049 3.408592 13  
## [32] {chocolate,   
## olive oil,   
## soup} => {mineral water} 0.001599787 0.8000000 0.001999733 3.356152 12  
## [33] {cooking oil,   
## eggs,   
## olive oil} => {mineral water} 0.001066524 0.8000000 0.001333156 3.356152 8  
## [34] {ground beef,   
## pancakes,   
## whole wheat rice} => {mineral water} 0.001333156 0.9090909 0.001466471 3.813809 10  
## [35] {burgers,   
## frozen vegetables,   
## low fat yogurt} => {mineral water} 0.001066524 0.8000000 0.001333156 3.356152 8  
## [36] {cake,   
## olive oil,   
## shrimp} => {mineral water} 0.001199840 1.0000000 0.001199840 4.195190 9  
## [37] {frozen vegetables,   
## olive oil,   
## shrimp} => {mineral water} 0.001866418 0.8235294 0.002266364 3.454862 14  
## [38] {burgers,   
## frozen vegetables,   
## olive oil} => {mineral water} 0.001199840 0.8181818 0.001466471 3.432428 9  
## [39] {frozen vegetables,   
## milk,   
## olive oil,   
## soup} => {mineral water} 0.001199840 0.8181818 0.001466471 3.432428 9  
## [40] {cake,   
## eggs,   
## milk,   
## turkey} => {mineral water} 0.001066524 0.8000000 0.001333156 3.356152 8  
## [41] {frozen vegetables,   
## milk,   
## spaghetti,   
## turkey} => {mineral water} 0.001199840 0.9000000 0.001333156 3.775671 9  
## [42] {frozen vegetables,   
## mineral water,   
## olive oil,   
## tomatoes} => {spaghetti} 0.001066524 0.8000000 0.001333156 4.594793 8  
## [43] {frozen vegetables,   
## ground beef,   
## mineral water,   
## tomatoes} => {spaghetti} 0.001199840 0.8181818 0.001466471 4.699220 9  
## [44] {chocolate,   
## frozen vegetables,   
## olive oil,   
## shrimp} => {mineral water} 0.001199840 0.9000000 0.001333156 3.775671 9  
## [45] {chocolate,   
## eggs,   
## milk,   
## olive oil} => {mineral water} 0.001066524 0.8000000 0.001333156 3.356152 8  
## [46] {chocolate,   
## french fries,   
## mineral water,   
## olive oil} => {spaghetti} 0.001066524 0.8000000 0.001333156 4.594793 8  
## [47] {chocolate,   
## eggs,   
## olive oil,   
## spaghetti} => {mineral water} 0.001199840 0.8181818 0.001466471 3.432428 9  
## [48] {chocolate,   
## frozen vegetables,   
## pancakes,   
## shrimp} => {mineral water} 0.001066524 0.8000000 0.001333156 3.356152 8  
## [49] {frozen vegetables,   
## ground beef,   
## mineral water,   
## shrimp} => {spaghetti} 0.001733102 0.8666667 0.001999733 4.977693 13  
## [50] {frozen vegetables,   
## milk,   
## shrimp,   
## spaghetti} => {mineral water} 0.001466471 0.8461538 0.001733102 3.549776 11  
## [51] {chocolate,   
## frozen vegetables,   
## shrimp,   
## spaghetti} => {mineral water} 0.001733102 0.8666667 0.001999733 3.635831 13  
## [52] {chocolate,   
## milk,   
## shrimp,   
## spaghetti} => {mineral water} 0.001199840 0.8181818 0.001466471 3.432428 9  
## [53] {french fries,   
## milk,   
## pancakes,   
## spaghetti} => {mineral water} 0.001066524 0.8000000 0.001333156 3.356152 8  
## [54] {chocolate,   
## eggs,   
## frozen vegetables,   
## ground beef} => {mineral water} 0.001466471 0.8461538 0.001733102 3.549776 11  
## [55] {chocolate,   
## ground beef,   
## milk,   
## mineral water,   
## spaghetti} => {frozen vegetables} 0.001066524 0.8888889 0.001199840 9.325253 8

rule 1 - frozen smoothier, spinach & mineral water rule 2 nonfat milk, turkey & mineral water

# 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”)

salerules<-sort(salerules, by="confidence", decreasing=TRUE)  
inspect(salerules[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