R-Association Rules

Vivian Bwana

2022-06-11

CREATING ASSOCIATION RULES

Defining the Question

a) Specifying the question

To create association rules that will identify relationships between different variables.

b) Metric for success

To be able to create association rules that will identify relationships between different variables.

c) Understanding the Context

Carrefour is one of the leading retail shops, (supermarkets) in the world. It was founded in France, in 1959. It has over the years expanded it's operations internationaly with the Kenyan branch opening in 1995.It has several branches in different parts of major cities countrywide.

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.

d) Experimental Design

1. Problem Definition
2. Data Sourcing
3. Check the Data
4. Perform Data Cleaning
5. Create Association Rules
6. Conclusion
7. Recommendation

e) Data Relevance /Sourcing

The dataset is relevant and reliable since it was provided by the client. We were able to draw relevant insights from it.

Data Understanding

Loading Libraries

```
# loading the necessary libraries
library(data.table)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(purrr)
```

```
##
## Attaching package: 'purrr'
```

```
## The following object is masked from 'package:caret':
##
##     lift
```

```
## The following object is masked from 'package:data.table':
##
##     transpose
```

```
library(dbplyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:dbplyr':
##
##     ident, sql
```

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

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

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

```
library(data.table)
library(arules)
```

```
## Loading required package: Matrix

##
## Attaching package: 'arules'

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

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

```
library(base)
library(tidyr)
```

```
##
## Attaching package: 'tidyr'

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

```
library(dplyr)
library(arules)
```

Part 3: Association Rules

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

Dataset for Part 3: Association Rules

```
# loading dataset path
path<-"~/Downloads/Supermarket_Sales_Dataset II.csv"

# reading transactions
caf_sales2<-read.transactions(path,sep = ',')
```

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

```
#preview
head(caf_sales2)
```

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

Exploring the Dataset

Dimensions

```
# checking the dimensions of the datasets
# to see how many rows and coulums there are
dim(caf_sales2)
```

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

There are 7501 and 119 columns in the dataset

Data Types

```
#Checking the datatypes of the dataset
str(caf_sales2)
```

```
## 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
```

The transactions dataset is inform of class with integers and characters

Column Names

```
# checking the column names
colnames(caf_sales2)
```

```
##   [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"
```

The columns are the list of items bought from the supermarket

Duplicates

```
# checking for duplicates
caf_sales2.duplicates <- caf_sales2[duplicated(caf_sales2),]

#printing duplicated rows
caf_sales2.duplicates
```

```
## transactions in sparse format with
##  2347 transactions (rows) and
##  119 items (columns)
```

There are no duplicates

CREATING ASSOCIATION RULES

Verifying object class

```
# Verifying the object's class
class(caf_sales2)
```

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

Previewing first 5 transactions

```
# Previewing our first 5 transactions
#
inspect(caf_sales2[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}
```

Preview items that make up dataset

```
# extracting a dataframe
items<-as.data.frame(itemLabels(caf_sales2))

# assigning column names
colnames(items) <- "Item"

# previewing
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
```

Previewing the last items of the dataframe

```r
# We use the tail function
tail(items, 10)
```

```
##                  Item
## 110            turkey
## 111    vegetables mix
## 112       water spray
## 113        white wine
## 114  whole weat flour
## 115 whole wheat pasta
## 116  whole wheat rice
## 117              yams
## 118       yogurt cake
## 119          zucchini
```

Descriptive Statistics Summary

```r
# checking summary of the dataframe
library(Hmisc)
```

```
## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:dplyr':
##
##     src, summarize

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```r
library(base)
#library(describe)
#describe(cafo12)
summary(caf_sales2)
```

```
## 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
```

The function summary gives the statistical summary The most purchased items include; Mineral water - 1788 Eggs - 1348 Spaghetti - 1306 French fries - 1282 Chocolate - 1229 Other items - 22405

Exploring the frequency

```r
# Exploring the frequency of some items
# from different ranges and performing
# the percentages they contribute in terms of the total transactions
# checking from the 20th to the 25th item
itemFrequency(caf_sales2[, 20:25],type = "absolute")
```

```
## cauliflower     cereals   champagne     chicken       chili   chocolate
##          36         193         351         450          46        1229
```

```r
round(itemFrequency(caf_sales2[, 20:25],type = "relative")*100,2)
```

```
## cauliflower     cereals   champagne     chicken       chili   chocolate
##        0.48        2.57        4.68        6.00        0.61       16.38
```

```r
print("________________________________________________________________________________")
```

```
## [1] "________________________________________________________________________________"
```

```r
# checking from the th to the 25th item
itemFrequency(caf_sales2[, 55:60],type = "absolute")
```

```
##      ground beef              gums              ham hand protein bar
##              737               101              199               39
##    herb & pepper             honey
##              371               356
```

```r
round(itemFrequency(caf_sales2[, 55:60],type = "relative")*100,2)
```

```
##      ground beef              gums              ham hand protein bar
##             9.83              1.35             2.65             0.52
##    herb & pepper             honey
##             4.95              4.75
```

Of the few items we have explored, chocolate has the highest percentage of 16.38%. The lowest was cauliflower with 0.48% of the total transactions.

Visualizing a chart of frequencies

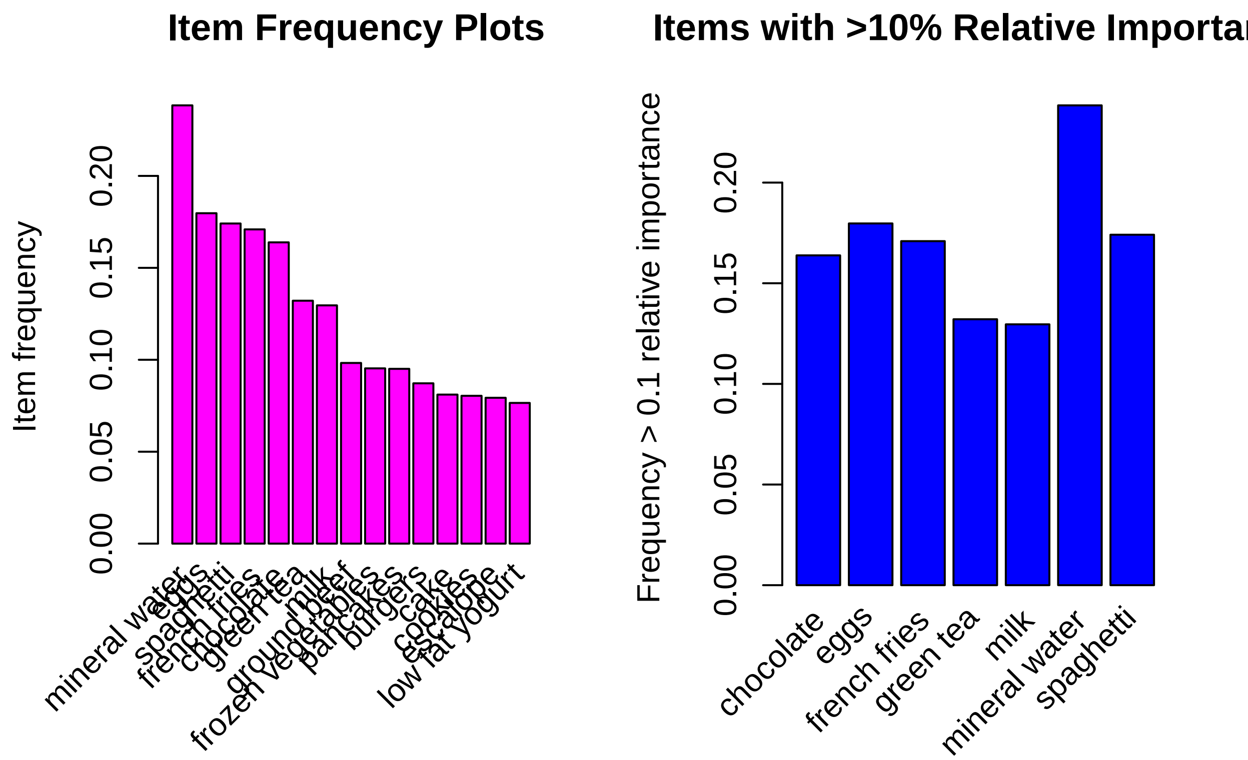
```r
# Producing a chart of frequencies and filtering
# to consider only items with above 10% of relative importance

# We will display top 15 most common items in the carrefour sales dataset

par(mfrow = c(1, 2))

# plot the frequency of items
# plotting top 30 items
itemFrequencyPlot(caf_sales2, topN = 15,col="magenta",ylab="Item frequency",main=" Item Frequency Plots

# plotting items that contribute atleast 10% of the total transactions
itemFrequencyPlot(caf_sales2, support = 0.1,col="blue",ylab="Frequency > 0.1 relative importance",
                  main=" Items with >10% Relative Importance")
```



The items that were most purchased in order are were: Mineral water, Eggs, Spaghetti, French fries, Chocolate, Green tea, milk, ground beef, frozen vegetables, pancakes, burgers, cake, cookies, escalope and low fat yogurt.

The items that had atleast 10% relative importance are 7: chocolate, eggs, french fries, green tea, milk, mineral water and spaghetti.

Model building

Building a model based on association rules

```
# Building a model based on association rules above
# we will use the apriori() function, Min Support as 0.001 and confidence as 0.8

rules1 <- apriori (caf_sales2, 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].
```

```
rules1
```

```
## set of 74 rules
```

Using 0.001 Min support and confidence as 0.8 we obtained 74 rules.

Tweaking the model We will change the values of minimum support and confidence level and see how the model performs

```
# Still using the apriori() function:

# Min Support as 0.002 and confidence as 0.8
rules2 <- apriori (caf_sales2, 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].
```

```
# Min Support as 0.001 and confidence as 0.6.
rules3 <- apriori (caf_sales2,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].
```

```r
# Min Support as 0.002 and confidence as 0.6.
rules4 <- apriori (caf_sales2, parameter = list(supp = 0.002, conf = 0.6))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.6    0.1    1 none FALSE            TRUE       5   0.002      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 ... [43 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
# printing
rules2
```

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

```
rules3
```

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

```
rules4
```

```
## set of 43 rules
```

rules2: set of 2 rules, rules3: set of 545 rules and rules4: set of 43 rules.

The results show that increasing the minimum support to 0.002 from 0.001 with a confidence level of 0.8 decreases the number of association rules from 74 to 2. This is too low to deduce much information from.

We also see that decreasing the confidence level to 0.6 from 0.8 while maintaining a minimum support of 0.001 increases the number of association rules from 74 to 545. These are too many and may not be useful.

Decreasing the confidence level to 0.6 from 0.8 and increasing the minimum support to 0.002 from 0.001 decreases the number of association rules from 74 to 43. This is a little moderate, however, we will maintain the first rules1 as it seems optimal.

```
# We check the properties of the model using the summary() function
summary(rules1)
```

Exploring the Model

```
## 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
##  caf_sales2          7501   0.001        0.8
##                                                                             call
##  apriori(data = caf_sales2, parameter = list(supp = 0.001, conf = 0.8))
```

Most rules have 4 and 5 items, with some having 3 and one having 6 items.

The function also gives us details such as minimum, maximum, median, mean, 1st and 3rd quantile information of the measures of support, confidence, coverage, lift and count.

```
# Observing rules built in our model
# first 10 model rules

inspect(rules1[1:10])
```

Randomly Inspecting the rules1 built in the model

```
##      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
## [6]  {milk, pasta}                 => {shrimp}        0.001599787 0.8571429
## [7]  {cooking oil, fromage blanc}  => {mineral water} 0.001199840 0.8181818
## [8]  {black tea, salmon}           => {mineral water} 0.001066524 0.8000000
## [9]  {black tea, frozen smoothie}  => {milk}          0.001199840 0.8181818
## [10] {red wine, tomato sauce}      => {chocolate}     0.001066524 0.8000000
##      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
## [6]  0.001866418 11.995203 12
## [7]  0.001466471  3.432428  9
## [8]  0.001333156  3.356152  8
## [9]  0.001466471  6.313973  9
## [10] 0.001333156  4.882669  8
```

From the first rule, we see that when someone buys frozen smoothie and/ spinach they are 88% most likely to buy mineral water.

The highest confidence level is 95%, which gives that when someone buys "mushroom cream sauce" and "pasta" then they are most likely to buy "escalope".

Ordering the rules by different criteria We order the rules by different criteria such as the level of confidence, by = "lift" or by = "support" and looking at the first 10 rules.

We will use confidence level for our analysis

```r
# sorting by confidence level and ordering by decreasing order
rules<-sort(rules1, by="confidence", decreasing=TRUE)
#printing the details
inspect(rules[1:10])
```

```
##      lhs                          rhs                 support confidence    coverage      lift count
## [1]  {french fries,
##       mushroom cream sauce,
##       pasta}                   => {escalope}      0.001066524  1.0000000 0.001066524 12.606723     8
## [2]  {ground beef,
##       light cream,
##       olive oil}               => {mineral water} 0.001199840  1.0000000 0.001199840  4.195190     9
## [3]  {cake,
##       meatballs,
##       mineral water}           => {milk}          0.001066524  1.0000000 0.001066524  7.717078     8
## [4]  {cake,
##       olive oil,
##       shrimp}                  => {mineral water} 0.001199840  1.0000000 0.001199840  4.195190     9
## [5]  {mushroom cream sauce,
##       pasta}                   => {escalope}      0.002532996  0.9500000 0.002666311 11.976387    19
## [6]  {red wine,
##       soup}                    => {mineral water} 0.001866418  0.9333333 0.001999733  3.915511    14
## [7]  {eggs,
##       mineral water,
##       pasta}                   => {shrimp}        0.001333156  0.9090909 0.001466471 12.722185    10
## [8]  {herb & pepper,
##       mineral water,
##       rice}                    => {ground beef}   0.001333156  0.9090909 0.001466471  9.252498    10
## [9]  {ground beef,
##       pancakes,
##       whole wheat rice}        => {mineral water} 0.001333156  0.9090909 0.001466471  3.813809    10
## [10] {frozen vegetables,
##       milk,
##       spaghetti,
##       turkey}                  => {mineral water} 0.001199840  0.9000000 0.001333156  3.775671     9
```

Rules 1-4 have 100% confidence. Rules 1: when someone buys french fries, mushroom cream sauce, pasta then they will buy escalope. Rules 2: when someone buys ground beef, light cream, olive oil then they will buy mineral water. Rules 3: when someone buys cake, meatballs, mineral water then they will buy milk. Rules 4: when someone buys cake, olive oil, shrimp then they will buy mineral water.

W e will create a subset of mineral water since it stands out and see how it affects the purchase of most items

Creating a subset that will give us the items that were bought before buying mineral water

```r
# Getting items purchased before mineral water
mineral_water_before <-subset(rules1,subset=rhs %pin% "mineral water")
# Sorting items by their confidence level
mineral_water_before_sorted<-sort(mineral_water_before,by="confidence",decreasing = TRUE)
# Viewing the top 10 items
inspect(mineral_water_before_sorted[1:10])
```

```
##      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
## [6]  {chocolate,
##       frozen vegetables,
##       olive oil,
##       shrimp}             => {mineral water} 0.001199840  0.9000000 0.001333156 3.775671     9
## [7]  {frozen smoothie,
##       spinach}            => {mineral water} 0.001066524  0.8888889 0.001199840 3.729058     8
## [8]  {cake,
##       meatballs,
##       milk}               => {mineral water} 0.001066524  0.8888889 0.001199840 3.729058     8
## [9]  {cake,
##       olive oil,
##       whole wheat pasta}  => {mineral water} 0.001066524  0.8888889 0.001199840 3.729058     8
## [10] {brownies,
##       eggs,
##       ground beef}        => {mineral water} 0.001066524  0.8888889 0.001199840 3.729058     8
```

People who purchase a combination of these items will most likely buy mineral water {ground beef, light cream, olive oil},{cake, olive oil, shrimp}, {red wine, soup}, {ground beef, pancakes, whole wheat rice}, {frozen vegetables, milk, spaghetti, turkey}, {chocolate, frozen vegetables, olive oil, shrimp}, {frozen smoothie, spinach},{cake, meatballs, milk}, {cake, olive oil, whole wheat pasta} and {brownies, eggs, ground beef}

Creating a subset that will give us the items that customers will most likely buy after buying mineral water

```r
# Getting items likely to be purchased after mineral water
mineral_water_after <-subset(rules1,subset=lhs %pin% "mineral water")
# Sorting items by their confidence level
mineral_water_after_sorted<-sort(mineral_water_after,by="confidence",decreasing = TRUE)
# Viewing the top 10 items
inspect(mineral_water_after_sorted[1:10])
```

```
##      lhs                        rhs                     support confidence    coverage      lift count
## [1]  {cake,
##       meatballs,
##       mineral water}      => {milk}              0.001066524  1.0000000 0.001066524  7.717078     8
## [2]  {eggs,
##       mineral water,
##       pasta}              => {shrimp}            0.001333156  0.9090909 0.001466471 12.722185    10
## [3]  {herb & pepper,
```

```
##       mineral water,
##       rice}              => {ground beef}       0.001333156  0.9090909 0.001466471  9.252498    10
## [4]  {light cream,
##       mineral water,
##       shrimp}            => {spaghetti}         0.001066524  0.8888889 0.001199840  5.105326     8
## [5]  {grated cheese,
##       mineral water,
##       rice}              => {ground beef}       0.001066524  0.8888889 0.001199840  9.046887     8
## [6]  {escalope,
##       hot dogs,
##       mineral water}     => {milk}              0.001066524  0.8888889 0.001199840  6.859625     8
## [7]  {chocolate,
##       ground beef,
##       milk,
##       mineral water,
##       spaghetti}         => {frozen vegetables} 0.001066524  0.8888889 0.001199840  9.325253     8
## [8]  {frozen vegetables,
##       ground beef,
##       mineral water,
##       shrimp}            => {spaghetti}         0.001733102  0.8666667 0.001999733  4.977693    13
## [9]  {mineral water,
##       pasta,
##       shrimp}            => {eggs}              0.001333156  0.8333333 0.001599787  4.637117    10
## [10] {frozen vegetables,
##       ground beef,
##       mineral water,
##       tomatoes}          => {spaghetti}         0.001199840  0.8181818 0.001466471  4.699220     9
```

People who purchase mineral water will most likely a combination of these items: Milk, shrimp, ground beef, spaghetti, frozen vegetables and eggs. The lists of item combinations of these purchases are shown above.

Conclusion

Mineral water seem to have the highest purchase influence in Carrefour supermarket.

Below is the distribution of these purchases: Mineral water - 1788, Eggs - 1348, Spaghetti - 1306, French fries - 1282, Chocolate - 1229, Other items - 22405. Other items that follow on the list are: Green tea, milk, ground beef, frozen vegetables, pancakes, burgers, cake, cookies, escalope and low fat yogurt.

The items that contribute highly in terms of the percentage of purchases include; chocolate, eggs, french fries, green tea, milk, mineral water and spaghetti.

We created different rules using different parameters and settled on the rule1 that obtained 74 sets of association rules.

We used this to create our model which was explored extensively. Different rules gave different confidence levels, we picked those that had 100% confidence levels: Rules 1: when someone buys french fries, mushroom cream sauce, pasta then they will buy escalope. Rules 2: when someone buys ground beef, light cream, olive oil then they will buy mineral water. Rules 3: when someone buys cake, meatballs, mineral water then they will buy milk. Rules 4: when someone buys cake, olive oil, shrimp then they will buy mineral water.

People who purchase a combination of these items will most likely buy mineral water {ground beef, light cream, olive oil},{cake, olive oil, shrimp}, {red wine, soup}, {ground beef, pancakes, whole wheat rice}, {frozen vegetables, milk, spaghetti, turkey}, {chocolate, frozen vegetables, olive oil, shrimp}, {frozen smoothie, spinach},{cake, meatballs, milk}, {cake, olive oil, whole wheat pasta} and {brownies, eggs, ground beef}

People who purchase mineral water will most likely a combination of these items: Milk, shrimp, ground beef, spaghetti, frozen vegetables and eggs. The lists of item combinations of these purchases are shown above.

Recommendation

Mineral water has the highest purchase influence in Carrefour supermarket. Other items that follow on the list are: Green tea, milk, ground beef, frozen vegetables, pancakes, burgers, cake, cookies, escalope and low fat yogurt. From the 74 association rules created, 4 had 100% confidence levels and hence it is recommended that these items be placed together: Rules 1: french fries, mushroom cream sauce, pasta include escalope. Rules 2: ground beef, light cream, olive oil include mineral water. Rules 3: cake, meatballs, mineral water include milk. Rules 4: cake, olive oil, shrimp include mineral water.

These items including the ones below should be in the same section of the supermarket: ground beef, light cream, olive oil, cake, shrimp, red wine, soup, pancakes, whole wheat rice, frozen vegetables, milk, spaghetti, turkey, chocolate, frozen smoothie, spinach, cake, meatballs, whole wheat pasta, brownies and eggs.