Association_analysis

Bill

11/13/2020

1. Problem Definition

1.1 Defining the Question

As a Data analyst at Carrefour Kenya,I have been consulted to undertake 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). My project will explore a recent marketing dataset provided by performing various unsupervised learning techniques and later providing recommendations based on my insights.

1.2 Specifying the Question

I am expected to identify relationships between variables in the dataset and to provide insights for my analysis.

1.4 Understanding the Context

Carrefour, one of the largest hypermarket chains in the world was introduced to the Middle East and North Africa (MENA) market in 1995 by Majid Al Futtaim, the leading shopping mall, retail and leisure pioneer across MENA.

Carrefour has become the most dynamic, fast-moving and exciting hypermarket chain in the region and shared its growth with more than 38,000 employees from more than 70 nationalities in 15 countries, providing shoppers with variety and value-for-money.

Carrefour ensures customer satisfaction and everyday convenience while offering unbeatable value for money with a vast array of more than 100,000 products, shoppers can purchase items for their every need, whether home electronics or fresh fruits from around the world, to locally produced items.

Carrefour opened its first outlet in Kenya in 2016, and currently operates over 250 hypermarkets, supermarkets, and online stores in 15 countries across the region, with plans to extend into 38 countries in the Middle East, Central Asia, Africa and Russia.

Carrefour always strive to provide the best quality and most diverse selection of household goods available in Kenya. Our value packs and combination discount offers means that we can offer these products at even lower costs, keeping your household essentials at unbeatable prices.

1.5 Experimental Design taken

1. Data Exploration
2. Data Cleaning and Formatting

3. Univariate Analysis
4. Bivariate Analysis
5. Multivariate Analysis
6. Implementing the solution through unsupervised machine learning,i.e. k-means, hierachical and DB-SCAN
7. Conclusion and Next steps # 2. Data Sourcing The data was availed to our data science team by the Carrefour's Sales and Marketing team therefore no data collection and scrapping was needed… We will just load our dataset in RStudio and begin the analysis process # 3. Check the Data

```
## Loading packages that we will use during our analysis
library("dplyr")
```

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

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

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

```
library("purrr")
library('tidyverse')
```

```
## -- Attaching packages ------------------------------------------------------------------------- tidy

## v ggplot2 3.3.2     v readr   1.3.1
## v tibble  3.0.2     v stringr 1.4.0
## v tidyr   1.1.0     v forcats 0.5.0

## -- Conflicts ---------------------------------------------------------------------------------- tidyverse_
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library('magrittr')
```

```
##
## Attaching package: 'magrittr'

## The following object is masked from 'package:tidyr':
##
##     extract

## The following object is masked from 'package:purrr':
##
##     set_names
```

```
library(’corrplot’)
```

```
## corrplot 0.84 loaded
```

```
library(’caret’)
```

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

```
##
## Attaching package: ’caret’
```

```
## The following object is masked from ’package:purrr’:
##
##     lift
```

```
library(’skimr’)
library(readr)
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
```

```
path <-"datasets/Supermarket_Sales_Dataset II.csv"

supermarket<-read.transactions(path, sep = ",")
```

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

```
supermarket
```

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

- Our dataset has 7501 transactions and 119 items

```r
# Verifying the object's class
# ---
# This should show us transactions as the type of data that we will need
# ---
#
class(supermarket)
```

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

```r
# Previewing our first 5 transactions
#
inspect(supermarket[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}
```

```r
# 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(supermarket)
```

```
## 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
```

Observations

1. Mineral water was the most purchased item with a total number of 1788
2. eggs were the next purchased item in the supermarket with a total number of 1348

3. Sphagetti, french fries and chocolate were the next most purchased items respectively

```r
# Exploring the frequency of some items in our supermarket
# i.e. goods ranging from 8 to 10 and performing
# some operation in percentage terms of the total transactions
#
itemFrequency(supermarket[, 8:10],type = "absolute")
```

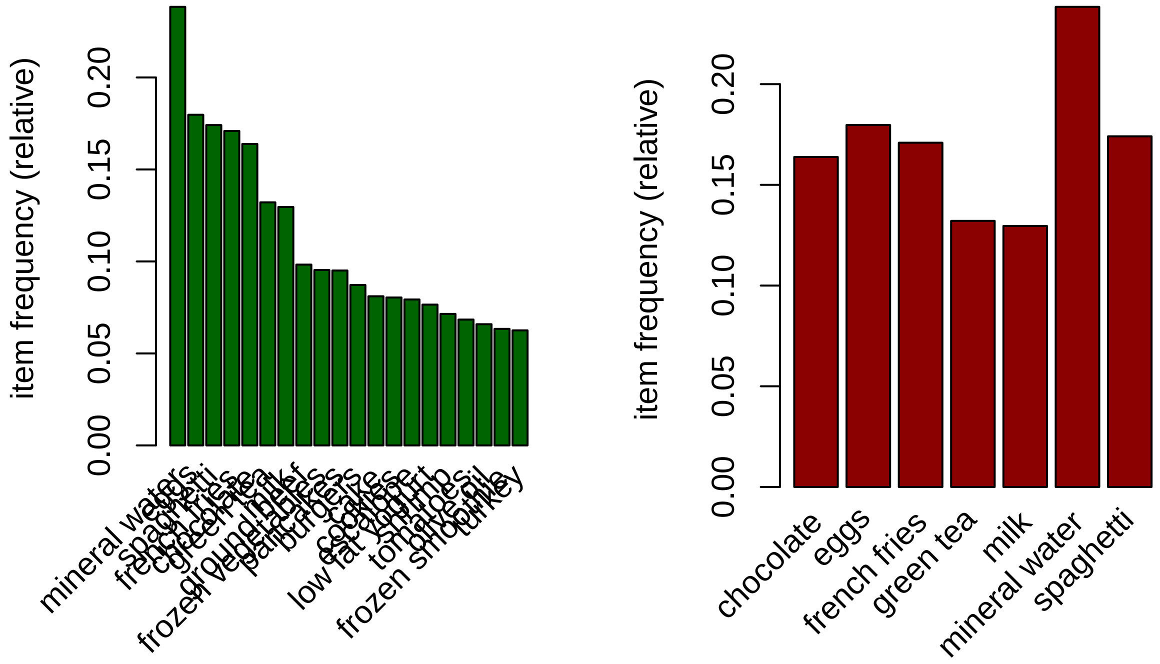
```
##   black tea blueberries  body spray 
##         107          69          86 
```

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

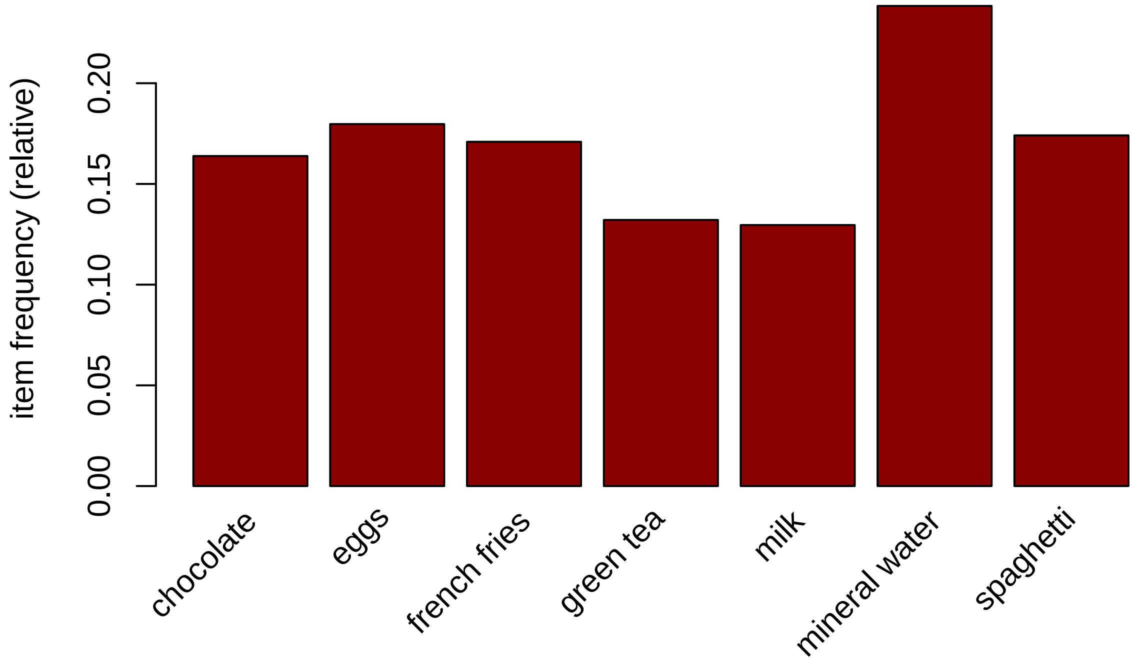
```
##   black tea blueberries  body spray 
##        1.43        0.92        1.15 
```

```
# Producing a chart of frequencies and fitering
# to consider only items with a minimum percentage
# of support/ considering a top x 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(supermarket, topN = 20,col="darkgreen")
itemFrequencyPlot(supermarket, support = 0.1,col="darkred")
```



```
itemFrequencyPlot(supermarket, 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 (supermarket, 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 [0.01s].
## 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 [0.02s].
## writing ... [74 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules
```

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

```
# We can perform an exploration of our model
# through the use of the summary function as shown
# ---
# Upon running the code, the function would give us information about the model
# i.e. the size of rules, depending on the items that contain these rules.
# More statistical information such as support, lift and confidence is also provided.
# ---
#
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
##  supermarket          7501   0.001        0.8
```

```
# Observing rules built in our model i.e. first 15 model rules
# ---
#
inspect(rules[1:15])
```

```
##      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
## [11] {pancakes,tomato sauce}       => {mineral water} 0.001066524 0.8000000
## [12] {chicken,protein bar}         => {spaghetti}     0.001199840 0.8181818
## [13] {meatballs,whole wheat pasta} => {milk}          0.001333156 0.8333333
## [14] {red wine,soup}               => {mineral water} 0.001866418 0.9333333
## [15] {turkey,whole wheat pasta}    => {mineral water} 0.001466471 0.8461538
##      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
## [11] 0.001333156  3.356152  8
## [12] 0.001466471  4.699220  9
## [13] 0.001599787  6.430898 10
## [14] 0.001999733  3.915511 14
## [15] 0.001733102  3.549776 11
```

##Observations -if someone buys frozen smoothie,spinach, there is an 88% confidence that he will buy mineral water - if someone buys bacon,pancakes, there is an 81% confidence that he will buy spaghetti - if someone buys mushroom cream sauce,pasta, there is an 95% confidence that he will buy escalope

```r
# 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[1:15])
```

```
##      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
## [11] {chocolate,
##       frozen vegetables,
##       olive oil,
##       shrimp}                => {mineral water} 0.001199840  0.9000000 0.001333156  3.775671     9
## [12] {frozen smoothie,
##       spinach}               => {mineral water} 0.001066524  0.8888889 0.001199840  3.729058     8
## [13] {black tea,
##       spaghetti,
##       turkey}                => {eggs}          0.001066524  0.8888889 0.001199840  4.946258     8
## [14] {light cream,
##       mineral water,
##       shrimp}                => {spaghetti}     0.001066524  0.8888889 0.001199840  5.105326     8
## [15] {cake,
##       meatballs,
##       milk}                  => {mineral water} 0.001066524  0.8888889 0.001199840  3.729058     8
```

```
# Interpretation
# ---
# The given five rules have a confidence of 100
# ---
```

```
library(arulesViz)
```

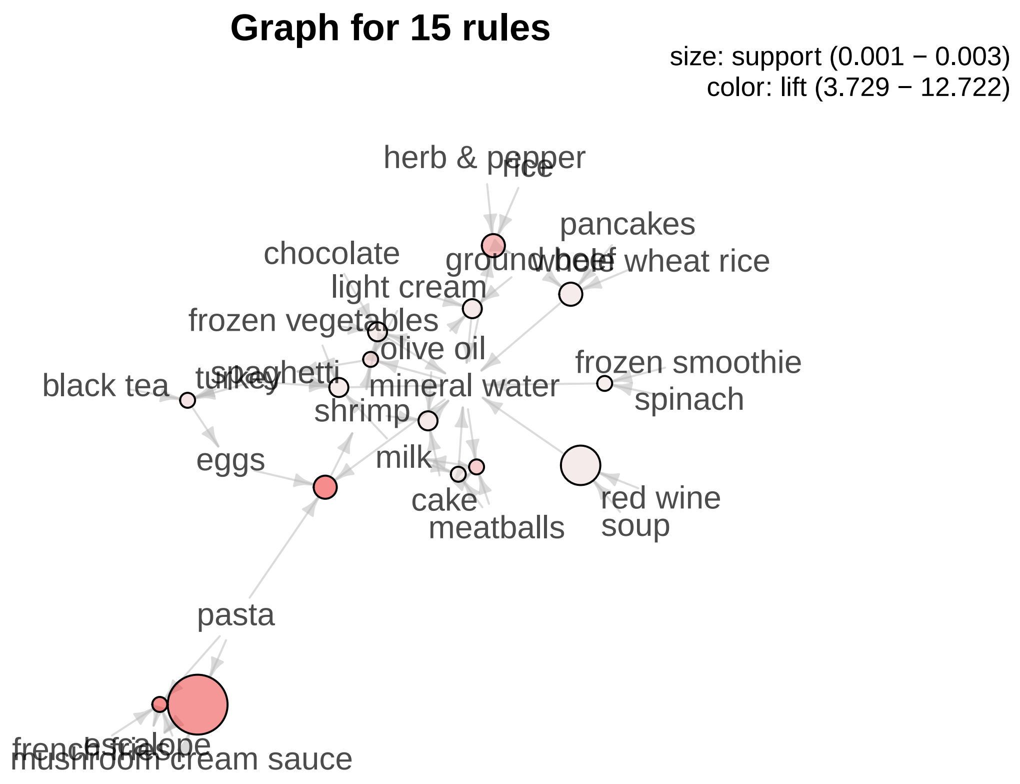
```
## Loading required package: grid
```

```
## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus
```

```
plot(rules[1:15],
method = "graph",
control = list(type = "items"))
```

```
## Warning: Unknown control parameters: type

## Available control parameters (with default values):
## main  =  Graph for 15 rules
## nodeColors    =  c("#66CC6680", "#9999CC80")
## nodeCol   =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF", "#EE0F0FFF", "#EE121
## edgeCol   =  c("#474747FF", "#494949FF", "#4B4B4BFF", "#4D4D4DFF", "#4F4F4FFF", "#515151FF", "#53535
## alpha     =  0.5
## cex   =  1
## itemLabels    =  TRUE
## labelCol  =  #000000B3
## measureLabels     =  FALSE
## precision     =  3
## layout    =  NULL
## layoutParams  =  list()
## arrowSize     =  0.5
## engine    =  igraph
## plot  =  TRUE
## plot_options  =  list()
## max   =  100
## verbose   =  FALSE
```



```
# 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
# ---
```

```r
# 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)
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
```

Observations

- ground beef, light cream, olive oil were bought before mineral water was purchased
- cake, olive oil, shrimp were also bought before mineral water was purchased
- red wine and soup were also bought before mineral water was purchased
- frozen vegetables, milk, spaghetti,turkey were also bought before mineral water was purchased

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

# Then order by confidence
mlk<-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
```

Observations

- cake,meatballs,mineral water, escalope,hot dogs, meatballs,whole wheat pasta, black tea,frozen smoothie and burgers,ground beef,olive oil were bought before milk

Insights

1. Mineral water was the most purchased item with a total number of 1788
2. eggs were the next purchased item in the supermarket with a total number of 1348

3. Sphagetti, french fries and chocolate were the next most purchased items respectively
4. Most people buy 3-5 items per transaction with majority being 4 items per transaction as per the rule length distribution
5. The following items should be placed near each other

- frozen smoothie,spinach
- red wine,tomato sauce
- black tea,frozen smoothie
- black tea,salmon
- cooking oil,fromage blanc
- milk,pasta
- mushroom cream sauce,pasta
- ground beef,nonfat milk
- bacon,pancakes
- nonfat milk,turkey
- turkey,whole wheat pasta
- red wine,soup
- cake,meatballs,mineral water
- chicken,protein bar
- meatballs,whole wheat pasta