Association Rules

kelvin njunge

9/10/2021

PROBLEM DEFINITION

a) Specifying the Question

create association rules that will allow you to identify relationships between variables in the dataset.

b) Defining the metrics for success

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 separate 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.

c) Understanding 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.

d) Recording the Experimental Design

Define the question, the metric for success, the context, experimental design taken. Read and explore the given dataset. create association rules that will allow you to identify relationships between variables in the dataset.

e) Relevance of the data

The data used for this project will inform the marketing department on the most relevant marketing strategies that will result in the highest no. of sales (total price including tax)

Data analysis

```
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 4.1.1
```

```
library(arulesViz)
```

```
## Warning: package 'arulesViz' was built under R version 4.1.1
```

```
## Loading required package: arules
```

```
## Warning: package 'arules' was built under R version 4.1.1
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

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

```
library(RColorBrewer)
library(ggplot2)
# Load libraries
library(tidyverse) # data manipulation
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
## v purrr   0.3.4
```

```
## Warning: package 'tibble' was built under R version 4.1.1
```

```
## Warning: package 'readr' was built under R version 4.1.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::arrange()   masks plyr::arrange()
## x purrr::compact()   masks plyr::compact()
## x dplyr::count()     masks plyr::count()
## x tidyr::expand()    masks Matrix::expand()
## x dplyr::failwith()  masks plyr::failwith()
## x dplyr::filter()    masks stats::filter()
## x dplyr::id()        masks plyr::id()
## x dplyr::lag()       masks stats::lag()
## x dplyr::mutate()    masks plyr::mutate()
## x tidyr::pack()      masks Matrix::pack()
## x dplyr::recode()    masks arules::recode()
## x dplyr::rename()    masks plyr::rename()
## x dplyr::summarise() masks plyr::summarise()
## x dplyr::summarize() masks plyr::summarize()
## x tidyr::unpack()    masks Matrix::unpack()
```

```
library(arules) # mining association rules and frequent itemsets
#library(arulesViz) # visualization techniques for association rules
library(knitr) #
library(gridExtra) # provides a number of user-level functions to work with "grid" graphics
```

```
## Warning: package 'gridExtra' was built under R version 4.1.1
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(lubridate) # work with dates and times
```

```
##
## Attaching package: 'lubridate'
```

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

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

```
getwd()
```

Loading dataset

```
## [1] "C:/Users/Ricky/Documents"
```

```
supermarket <-read.transactions("C:\\Users\\Ricky\\Documents\\Supermarket_Sales_Dataset II.csv",sep = "
```

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

Data Exploratory

```
# verify object class
class(supermarket)
```

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

```
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
```

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

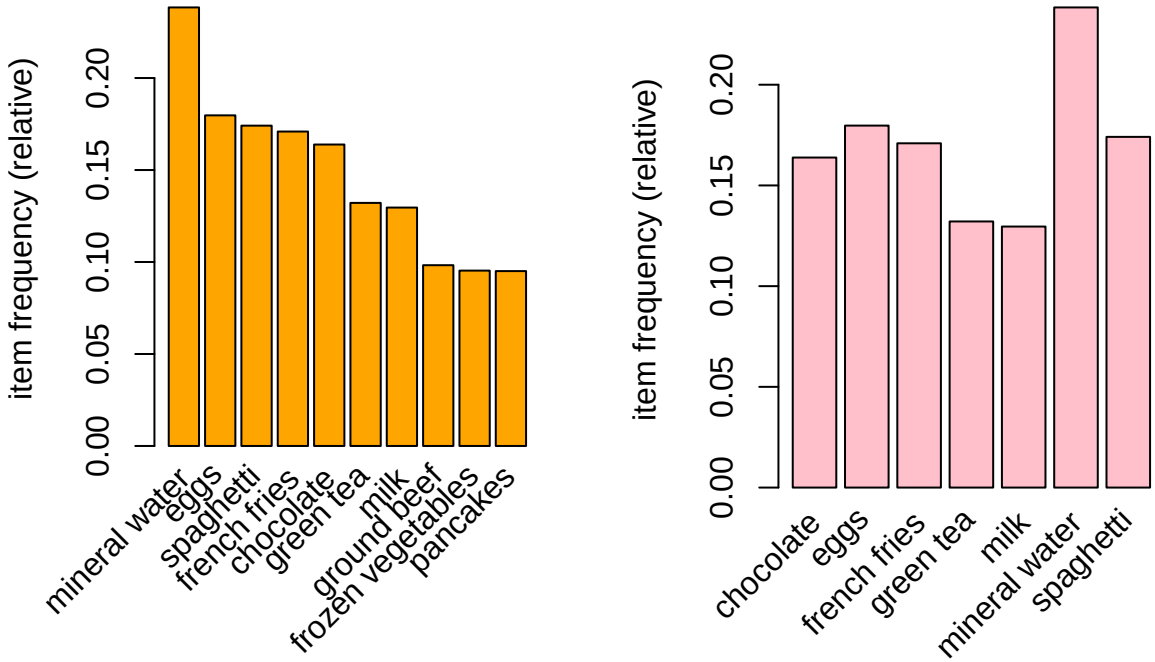
```
##     items
## [1] {almonds,
##      antioxydant juice,
##      avocado,
##      cottage cheese,
##      energy drink,
##      frozen smoothie,
##      green grapes,
##      green tea,
##      honey,
##      low fat yogurt,
##      mineral water,
##      olive oil,
##      salad,
##      salmon,
##      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}
```

```
# frequency
itemFrequency(supermarket[,1:3])
```

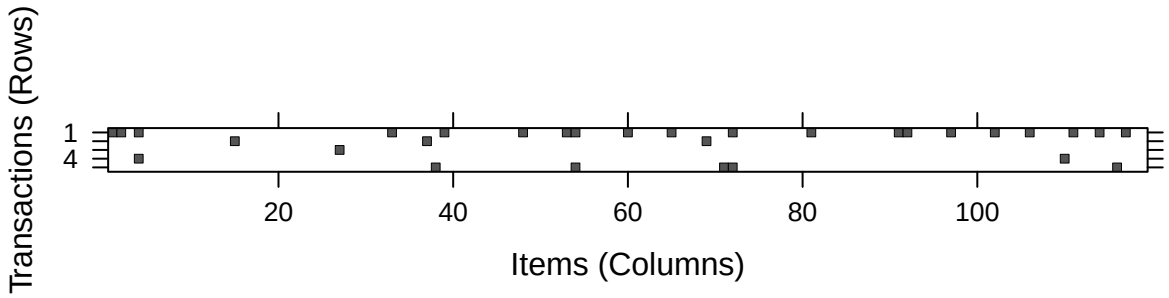
```
##           almonds antioxydant juice         asparagus
##       0.020397280       0.008932142       0.004799360
```

```
# 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 = 10,col="orange")
itemFrequencyPlot(supermarket, support = 0.1,col="pink")
```

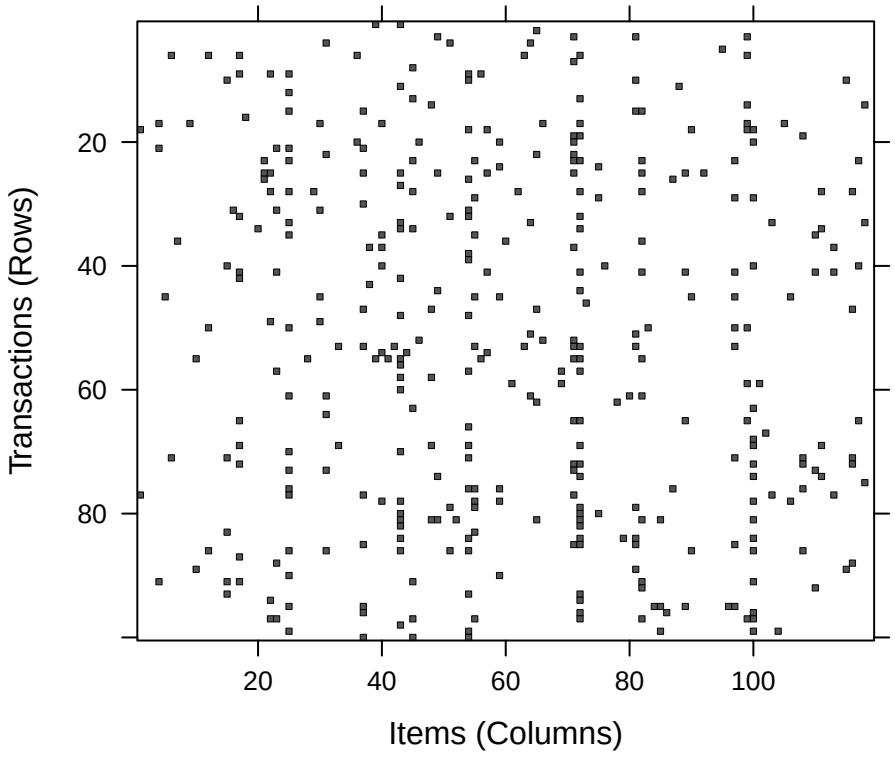


Here you can see that top 10 most common items in the transactions dataset and the items whose relative importance is at least 10% are mineral water,eggs,spaghetti,french fries,chocolate,green tea,milk

```
# a visualization of the sparse matrix for the first five transactions
image(supermarket[1:5])
```




```
# visualization of a random sample of 100 transactions
image(sample(supermarket, 100))
```




```
#Training a model on the data
library(arules)
# default settings result in zero rules learned
apriori(supermarket)
```

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

```
# Building a model based on association rules
# using the apriori function
# set better support and confidence levels to learn more rules
# We use Min Support as 0.006 and confidence as 0.25
# The minlen defines the minimum number of items in each itemset of frequent items which we use only 2.
Supermarket_rules <- apriori(supermarket, parameter = list(support =0.006, confidence = 0.25, minlen = 2
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.25    0.1    1 none FALSE            TRUE       5   0.006      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: 45
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.01s].
## sorting and recoding items ... [97 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [272 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
Supermarket_rules
```

```
## set of 272 rules
```

Here we can see using a support of 0.006, confidence of 0.25 and minlen of 2 we obtain a set of 272 rules.

```
## inspecting the first three rules
inspect(Supermarket_rules[1:3])
```

```
##     lhs                rhs              support     confidence coverage
## [1] {tomato sauce} => {spaghetti}     0.006265831 0.4433962  0.01413145
## [2] {light cream}  => {mineral water} 0.007332356 0.4700855  0.01559792
## [3] {protein bar}  => {mineral water} 0.007732302 0.4172662  0.01853086
##     lift     count
## [1] 2.546642 47
## [2] 1.972098 55
## [3] 1.750511 58
```

- If someone buys tomato sauce they are 44% likely to buy spaghetti too
- If someone buys light cream they are 47% likely to buy mineral water

- If someone buys protein bar they are 42% likely to buy mineral water.

```
# Improving model performance
# sorting grocery rules by confidence
# then looking at the first five rules.
inspect(sort(Supermarket_rules, by = "confidence", decreasing=TRUE)[1:5])
```

```
##     lhs                                rhs             support     confidence
## [1] {frozen vegetables,olive oil}   => {mineral water} 0.006532462 0.5764706
## [2] {milk,soup}                     => {mineral water} 0.008532196 0.5614035
## [3] {cooking oil,eggs}              => {mineral water} 0.006399147 0.5454545
## [4] {frozen vegetables,ground beef} => {mineral water} 0.009198773 0.5433071
## [5] {milk,turkey}                   => {mineral water} 0.006132516 0.5411765
##     coverage   lift     count
## [1] 0.01133182 2.418404 49
## [2] 0.01519797 2.355194 64
## [3] 0.01173177 2.288286 48
## [4] 0.01693108 2.279277 69
## [5] 0.01133182 2.270338 46
```

The first rule has a confidence of 58,the second rule has a confidence of 56 the third rule has a cofidence of 55 the fourth rule has a confidence of 54 and the fifth rule has a confidence of 54.

```
# finding subsets of rules containing any herb & pepper
herb_pepper_rules <- subset(Supermarket_rules, items %in% "herb & pepper")
inspect(herb_pepper_rules)
```

```
##     lhs                              rhs             support     confidence
## [1] {herb & pepper}               => {ground beef}   0.015997867 0.3234501
## [2] {herb & pepper}               => {eggs}          0.012531662 0.2533693
## [3] {herb & pepper}               => {spaghetti}     0.016264498 0.3288410
## [4] {herb & pepper}               => {mineral water} 0.017064391 0.3450135
## [5] {ground beef,herb & pepper}   => {spaghetti}     0.006399147 0.4000000
## [6] {herb & pepper,spaghetti}     => {ground beef}   0.006399147 0.3934426
## [7] {ground beef,herb & pepper}   => {mineral water} 0.006665778 0.4166667
## [8] {herb & pepper,mineral water} => {ground beef}   0.006665778 0.3906250
##     coverage   lift     count
## [1] 0.04946007 3.291994 120
## [2] 0.04946007 1.409883  94
## [3] 0.04946007 1.888695 122
## [4] 0.04946007 1.447397 128
## [5] 0.01599787 2.297397  48
## [6] 0.01626450 4.004360  48
## [7] 0.01599787 1.747996  50
## [8] 0.01706439 3.975683  50
```

```
# writing the rules to a CSV file
write(Supermarket_rules, file = "Supermarket_rules.csv",
      sep = ",", quote = TRUE, row.names = FALSE)
# converting the rule set to a data frame
Supermarket_rules_df <- as(Supermarket_rules, "data.frame")
# The rules identified by the algorithms
str(Supermarket_rules_df)
```

```
## 'data.frame':    272 obs. of  6 variables:
##  $ rules     : chr  "{tomato sauce} => {spaghetti}" "{light cream} => {mineral water}" "{protein bar
##  $ support   : num  0.00627 0.00733 0.00773 0.0064 0.00773 ...
##  $ confidence: num  0.443 0.47 0.417 0.306 0.369 ...
##  $ coverage  : num  0.0141 0.0156 0.0185 0.0209 0.0209 ...
##  $ lift      : num  2.55 1.97 1.75 1.76 1.55 ...
##  $ count     : int  47 55 58 48 58 51 58 51 49 57 ...
```

```
Supermarket_rules_df$rules
```

```
##   [1] "{tomato sauce} => {spaghetti}"
##   [2] "{light cream} => {mineral water}"
##   [3] "{protein bar} => {mineral water}"
##   [4] "{meatballs} => {spaghetti}"
##   [5] "{meatballs} => {mineral water}"
##   [6] "{energy bar} => {mineral water}"
##   [7] "{rice} => {mineral water}"
##   [8] "{parmesan cheese} => {spaghetti}"
##   [9] "{almonds} => {eggs}"
##  [10] "{almonds} => {mineral water}"
##  [11] "{light mayo} => {french fries}"
##  [12] "{vegetables mix} => {mineral water}"
##  [13] "{french wine} => {spaghetti}"
##  [14] "{french wine} => {mineral water}"
##  [15] "{oil} => {chocolate}"
##  [16] "{oil} => {mineral water}"
##  [17] "{fresh tuna} => {spaghetti}"
##  [18] "{fresh tuna} => {mineral water}"
##  [19] "{pepper} => {spaghetti}"
##  [20] "{pepper} => {mineral water}"
##  [21] "{ham} => {spaghetti}"
##  [22] "{ham} => {mineral water}"
##  [23] "{cereals} => {green tea}"
##  [24] "{cereals} => {milk}"
##  [25] "{cereals} => {spaghetti}"
##  [26] "{cereals} => {mineral water}"
##  [27] "{red wine} => {eggs}"
##  [28] "{red wine} => {spaghetti}"
##  [29] "{red wine} => {mineral water}"
##  [30] "{butter} => {chocolate}"
##  [31] "{whole wheat pasta} => {olive oil}"
##  [32] "{whole wheat pasta} => {milk}"
##  [33] "{whole wheat pasta} => {spaghetti}"
##  [34] "{whole wheat pasta} => {mineral water}"
##  [35] "{cottage cheese} => {mineral water}"
##  [36] "{hot dogs} => {spaghetti}"
##  [37] "{hot dogs} => {mineral water}"
##  [38] "{tomato juice} => {mineral water}"
##  [39] "{brownies} => {mineral water}"
##  [40] "{avocado} => {mineral water}"
##  [41] "{fresh bread} => {mineral water}"
##  [42] "{salmon} => {chocolate}"
##  [43] "{salmon} => {spaghetti}"
##  [44] "{salmon} => {mineral water}"
```

```
##  [45] "{honey} => {spaghetti}"
##  [46] "{honey} => {mineral water}"
##  [47] "{herb & pepper} => {ground beef}"
##  [48] "{herb & pepper} => {eggs}"
##  [49] "{herb & pepper} => {spaghetti}"
##  [50] "{herb & pepper} => {mineral water}"
##  [51] "{grated cheese} => {spaghetti}"
##  [52] "{grated cheese} => {mineral water}"
##  [53] "{soup} => {milk}"
##  [54] "{soup} => {spaghetti}"
##  [55] "{soup} => {mineral water}"
##  [56] "{cooking oil} => {chocolate}"
##  [57] "{cooking oil} => {spaghetti}"
##  [58] "{cooking oil} => {mineral water}"
##  [59] "{whole wheat rice} => {mineral water}"
##  [60] "{turkey} => {eggs}"
##  [61] "{turkey} => {spaghetti}"
##  [62] "{turkey} => {mineral water}"
##  [63] "{chicken} => {spaghetti}"
##  [64] "{chicken} => {mineral water}"
##  [65] "{frozen smoothie} => {mineral water}"
##  [66] "{low fat yogurt} => {mineral water}"
##  [67] "{tomatoes} => {spaghetti}"
##  [68] "{tomatoes} => {mineral water}"
##  [69] "{olive oil} => {milk}"
##  [70] "{olive oil} => {spaghetti}"
##  [71] "{olive oil} => {mineral water}"
##  [72] "{shrimp} => {chocolate}"
##  [73] "{shrimp} => {spaghetti}"
##  [74] "{shrimp} => {mineral water}"
##  [75] "{cake} => {mineral water}"
##  [76] "{burgers} => {french fries}"
##  [77] "{burgers} => {eggs}"
##  [78] "{burgers} => {mineral water}"
##  [79] "{pancakes} => {spaghetti}"
##  [80] "{pancakes} => {mineral water}"
##  [81] "{frozen vegetables} => {spaghetti}"
##  [82] "{frozen vegetables} => {mineral water}"
##  [83] "{ground beef} => {spaghetti}"
##  [84] "{ground beef} => {mineral water}"
##  [85] "{milk} => {spaghetti}"
##  [86] "{milk} => {mineral water}"
##  [87] "{chocolate} => {mineral water}"
##  [88] "{eggs} => {mineral water}"
##  [89] "{spaghetti} => {mineral water}"
##  [90] "{mineral water} => {spaghetti}"
##  [91] "{salmon,spaghetti} => {mineral water}"
##  [92] "{mineral water,salmon} => {spaghetti}"
##  [93] "{ground beef,herb & pepper} => {spaghetti}"
##  [94] "{herb & pepper,spaghetti} => {ground beef}"
##  [95] "{ground beef,herb & pepper} => {mineral water}"
##  [96] "{herb & pepper,mineral water} => {ground beef}"
##  [97] "{grated cheese,spaghetti} => {mineral water}"
##  [98] "{grated cheese,mineral water} => {spaghetti}"
```

```
##  [99] "{milk,soup} => {mineral water}"
## [100] "{mineral water,soup} => {milk}"
## [101] "{soup,spaghetti} => {mineral water}"
## [102] "{mineral water,soup} => {spaghetti}"
## [103] "{cooking oil,eggs} => {mineral water}"
## [104] "{cooking oil,mineral water} => {eggs}"
## [105] "{cooking oil,spaghetti} => {mineral water}"
## [106] "{cooking oil,mineral water} => {spaghetti}"
## [107] "{spaghetti,whole wheat rice} => {mineral water}"
## [108] "{mineral water,whole wheat rice} => {spaghetti}"
## [109] "{milk,turkey} => {mineral water}"
## [110] "{mineral water,turkey} => {milk}"
## [111] "{spaghetti,turkey} => {mineral water}"
## [112] "{mineral water,turkey} => {spaghetti}"
## [113] "{chicken,milk} => {mineral water}"
## [114] "{chicken,mineral water} => {milk}"
## [115] "{chicken,chocolate} => {mineral water}"
## [116] "{chicken,mineral water} => {chocolate}"
## [117] "{chicken,spaghetti} => {mineral water}"
## [118] "{chicken,mineral water} => {spaghetti}"
## [119] "{frozen smoothie,milk} => {mineral water}"
## [120] "{frozen smoothie,mineral water} => {milk}"
## [121] "{frozen smoothie,spaghetti} => {mineral water}"
## [122] "{frozen smoothie,mineral water} => {spaghetti}"
## [123] "{eggs,low fat yogurt} => {mineral water}"
## [124] "{low fat yogurt,mineral water} => {eggs}"
## [125] "{frozen vegetables,tomatoes} => {spaghetti}"
## [126] "{spaghetti,tomatoes} => {frozen vegetables}"
## [127] "{milk,tomatoes} => {mineral water}"
## [128] "{mineral water,tomatoes} => {milk}"
## [129] "{spaghetti,tomatoes} => {mineral water}"
## [130] "{mineral water,tomatoes} => {spaghetti}"
## [131] "{frozen vegetables,olive oil} => {mineral water}"
## [132] "{ground beef,olive oil} => {spaghetti}"
## [133] "{olive oil,spaghetti} => {ground beef}"
## [134] "{ground beef,olive oil} => {mineral water}"
## [135] "{milk,olive oil} => {spaghetti}"
## [136] "{olive oil,spaghetti} => {milk}"
## [137] "{milk,olive oil} => {mineral water}"
## [138] "{mineral water,olive oil} => {milk}"
## [139] "{chocolate,olive oil} => {spaghetti}"
## [140] "{olive oil,spaghetti} => {chocolate}"
## [141] "{chocolate,olive oil} => {mineral water}"
## [142] "{mineral water,olive oil} => {chocolate}"
## [143] "{olive oil,spaghetti} => {mineral water}"
## [144] "{mineral water,olive oil} => {spaghetti}"
## [145] "{frozen vegetables,shrimp} => {mineral water}"
## [146] "{mineral water,shrimp} => {frozen vegetables}"
## [147] "{milk,shrimp} => {mineral water}"
## [148] "{mineral water,shrimp} => {milk}"
## [149] "{chocolate,shrimp} => {spaghetti}"
## [150] "{shrimp,spaghetti} => {chocolate}"
## [151] "{chocolate,shrimp} => {mineral water}"
## [152] "{mineral water,shrimp} => {chocolate}"
```

```
## [153] "{shrimp,spaghetti} => {mineral water}"
## [154] "{mineral water,shrimp} => {spaghetti}"
## [155] "{cake,milk} => {mineral water}"
## [156] "{cake,french fries} => {mineral water}"
## [157] "{cake,eggs} => {mineral water}"
## [158] "{cake,mineral water} => {eggs}"
## [159] "{cake,spaghetti} => {mineral water}"
## [160] "{cake,mineral water} => {spaghetti}"
## [161] "{burgers,milk} => {spaghetti}"
## [162] "{burgers,spaghetti} => {milk}"
## [163] "{burgers,milk} => {mineral water}"
## [164] "{burgers,mineral water} => {milk}"
## [165] "{burgers,french fries} => {eggs}"
## [166] "{burgers,eggs} => {french fries}"
## [167] "{burgers,chocolate} => {spaghetti}"
## [168] "{burgers,spaghetti} => {chocolate}"
## [169] "{burgers,eggs} => {spaghetti}"
## [170] "{burgers,spaghetti} => {eggs}"
## [171] "{burgers,eggs} => {mineral water}"
## [172] "{burgers,mineral water} => {eggs}"
## [173] "{burgers,spaghetti} => {mineral water}"
## [174] "{burgers,mineral water} => {spaghetti}"
## [175] "{frozen vegetables,pancakes} => {mineral water}"
## [176] "{ground beef,pancakes} => {spaghetti}"
## [177] "{pancakes,spaghetti} => {ground beef}"
## [178] "{ground beef,pancakes} => {mineral water}"
## [179] "{milk,pancakes} => {mineral water}"
## [180] "{french fries,pancakes} => {mineral water}"
## [181] "{chocolate,pancakes} => {spaghetti}"
## [182] "{pancakes,spaghetti} => {chocolate}"
## [183] "{chocolate,pancakes} => {mineral water}"
## [184] "{mineral water,pancakes} => {chocolate}"
## [185] "{eggs,pancakes} => {spaghetti}"
## [186] "{pancakes,spaghetti} => {eggs}"
## [187] "{eggs,pancakes} => {mineral water}"
## [188] "{pancakes,spaghetti} => {mineral water}"
## [189] "{mineral water,pancakes} => {spaghetti}"
## [190] "{frozen vegetables,ground beef} => {spaghetti}"
## [191] "{frozen vegetables,spaghetti} => {ground beef}"
## [192] "{frozen vegetables,ground beef} => {mineral water}"
## [193] "{frozen vegetables,mineral water} => {ground beef}"
## [194] "{frozen vegetables,milk} => {chocolate}"
## [195] "{chocolate,frozen vegetables} => {milk}"
## [196] "{frozen vegetables,milk} => {eggs}"
## [197] "{eggs,frozen vegetables} => {milk}"
## [198] "{frozen vegetables,milk} => {spaghetti}"
## [199] "{frozen vegetables,spaghetti} => {milk}"
## [200] "{frozen vegetables,milk} => {mineral water}"
## [201] "{frozen vegetables,mineral water} => {milk}"
## [202] "{french fries,frozen vegetables} => {mineral water}"
## [203] "{chocolate,frozen vegetables} => {spaghetti}"
## [204] "{frozen vegetables,spaghetti} => {chocolate}"
## [205] "{chocolate,frozen vegetables} => {mineral water}"
## [206] "{frozen vegetables,mineral water} => {chocolate}"
```

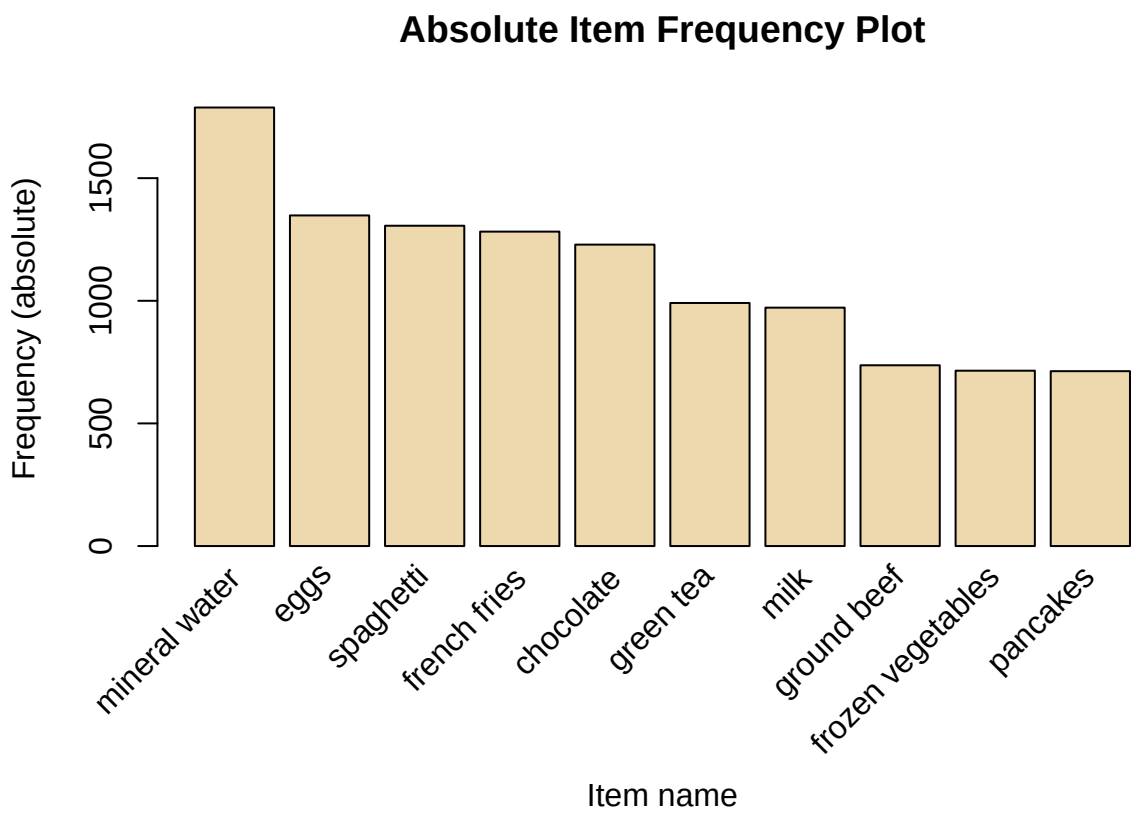
```
## [207] "{eggs,frozen vegetables} => {mineral water}"
## [208] "{frozen vegetables,mineral water} => {eggs}"
## [209] "{frozen vegetables,spaghetti} => {mineral water}"
## [210] "{frozen vegetables,mineral water} => {spaghetti}"
## [211] "{green tea,ground beef} => {spaghetti}"
## [212] "{ground beef,milk} => {chocolate}"
## [213] "{chocolate,ground beef} => {milk}"
## [214] "{ground beef,milk} => {spaghetti}"
## [215] "{milk,spaghetti} => {ground beef}"
## [216] "{ground beef,milk} => {mineral water}"
## [217] "{ground beef,mineral water} => {milk}"
## [218] "{chocolate,ground beef} => {eggs}"
## [219] "{eggs,ground beef} => {chocolate}"
## [220] "{chocolate,ground beef} => {spaghetti}"
## [221] "{chocolate,ground beef} => {mineral water}"
## [222] "{ground beef,mineral water} => {chocolate}"
## [223] "{eggs,ground beef} => {spaghetti}"
## [224] "{eggs,ground beef} => {mineral water}"
## [225] "{ground beef,spaghetti} => {mineral water}"
## [226] "{ground beef,mineral water} => {spaghetti}"
## [227] "{mineral water,spaghetti} => {ground beef}"
## [228] "{eggs,green tea} => {french fries}"
## [229] "{chocolate,green tea} => {spaghetti}"
## [230] "{green tea,spaghetti} => {chocolate}"
## [231] "{chocolate,green tea} => {mineral water}"
## [232] "{eggs,green tea} => {mineral water}"
## [233] "{green tea,spaghetti} => {mineral water}"
## [234] "{green tea,mineral water} => {spaghetti}"
## [235] "{french fries,milk} => {chocolate}"
## [236] "{french fries,milk} => {eggs}"
## [237] "{french fries,milk} => {spaghetti}"
## [238] "{french fries,milk} => {mineral water}"
## [239] "{chocolate,milk} => {eggs}"
## [240] "{eggs,milk} => {chocolate}"
## [241] "{chocolate,eggs} => {milk}"
## [242] "{chocolate,milk} => {spaghetti}"
## [243] "{milk,spaghetti} => {chocolate}"
## [244] "{chocolate,spaghetti} => {milk}"
## [245] "{chocolate,milk} => {mineral water}"
## [246] "{milk,mineral water} => {chocolate}"
## [247] "{chocolate,mineral water} => {milk}"
## [248] "{eggs,milk} => {spaghetti}"
## [249] "{milk,spaghetti} => {eggs}"
## [250] "{eggs,milk} => {mineral water}"
## [251] "{milk,mineral water} => {eggs}"
## [252] "{eggs,mineral water} => {milk}"
## [253] "{milk,spaghetti} => {mineral water}"
## [254] "{milk,mineral water} => {spaghetti}"
## [255] "{mineral water,spaghetti} => {milk}"
## [256] "{chocolate,eggs} => {french fries}"
## [257] "{french fries,spaghetti} => {chocolate}"
## [258] "{french fries,mineral water} => {chocolate}"
## [259] "{french fries,spaghetti} => {eggs}"
## [260] "{french fries,spaghetti} => {mineral water}"
```

```
## [261] "{french fries,mineral water} => {spaghetti}"
## [262] "{chocolate,eggs} => {spaghetti}"
## [263] "{chocolate,spaghetti} => {eggs}"
## [264] "{eggs,spaghetti} => {chocolate}"
## [265] "{chocolate,eggs} => {mineral water}"
## [266] "{chocolate,mineral water} => {eggs}"
## [267] "{eggs,mineral water} => {chocolate}"
## [268] "{chocolate,spaghetti} => {mineral water}"
## [269] "{chocolate,mineral water} => {spaghetti}"
## [270] "{mineral water,spaghetti} => {chocolate}"
## [271] "{eggs,spaghetti} => {mineral water}"
## [272] "{eggs,mineral water} => {spaghetti}"
```

```
class(supermarket)
```

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

```
# Absolute Item Frequency Plot
itemFrequencyPlot(supermarket, topN=10, type="absolute", col="wheat2",xlab="Item name",
                  ylab="Frequency (absolute)", main="Absolute Item Frequency Plot")
```



```
# Generating a summary of the transaction dataset
#  information such as the most purchased items,
# distribution of the item sets (no. of items purchased in each transaction),
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
```

```
install.packages("RColorBrewer")
```

```
## Warning: package 'RColorBrewer' is in use and will not be installed
```

```
#include library RColorBrewer
library(RColorBrewer)
```

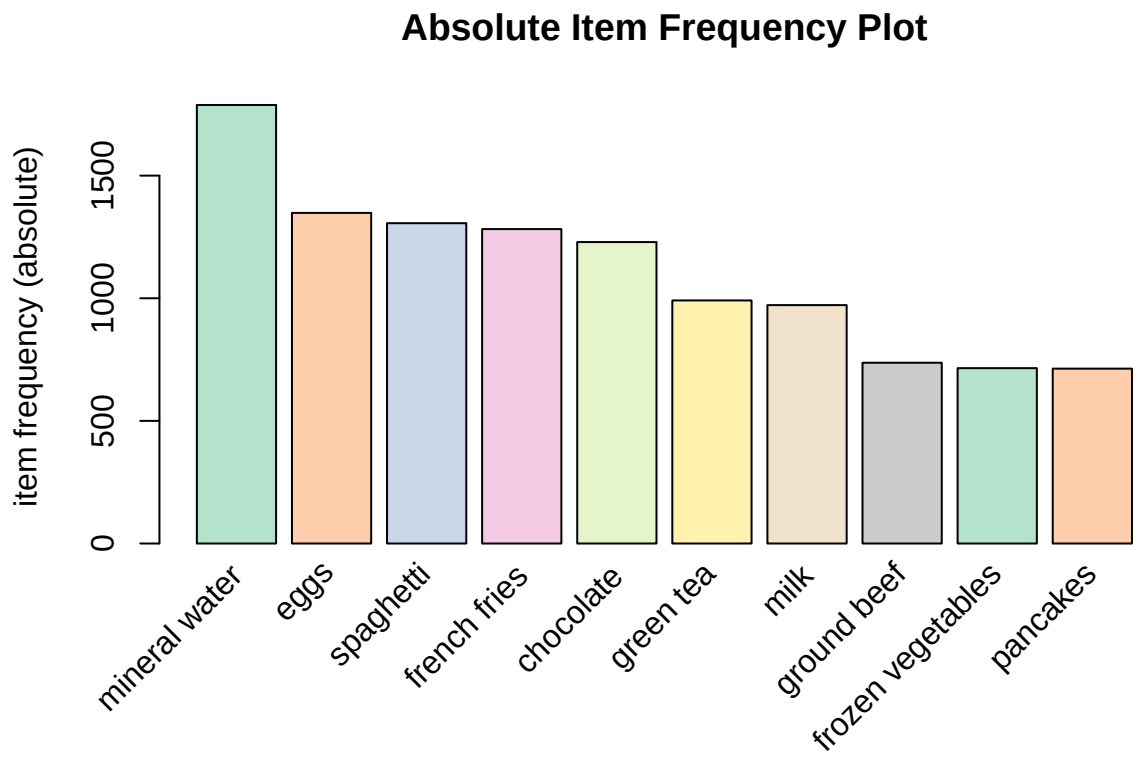
```
itemFrequency(supermarket)
```

```
##              almonds    antioxydant juice           asparagus
##         0.0203972804         0.0089321424         0.0047993601
##              avocado          babies food                bacon
##         0.0333288895         0.0045327290         0.0086655113
##       barbecue sauce            black tea          blueberries
##         0.0107985602         0.0142647647         0.0091987735
##           body spray              bramble             brownies
##         0.0114651380         0.0018664178         0.0337288362
##            bug spray         burger sauce              burgers
##         0.0086655113         0.0058658845         0.0871883749
##               butter                 cake           candy bars
```

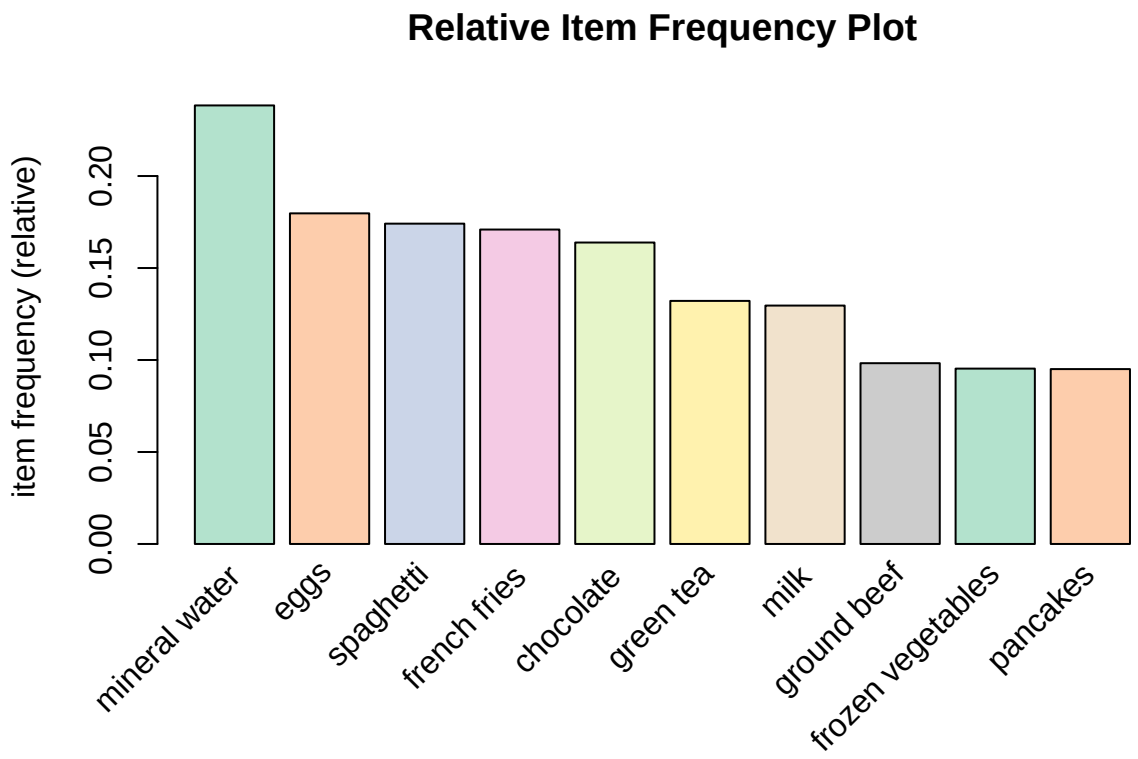
```
##         0.0301293161         0.0810558592         0.0097320357
##              carrots          cauliflower              cereals
##         0.0153312892         0.0047993601         0.0257299027
##            champagne              chicken                chili
##         0.0467937608         0.0599920011         0.0061325157
##            chocolate      chocolate bread              chutney
##         0.1638448207         0.0042660979         0.0041327823
##                cider  clothes accessories              cookies
##         0.0105319291         0.0083988801         0.0803892814
##          cooking oil                 corn       cottage cheese
##         0.0510598587         0.0047993601         0.0318624183
##                cream         dessert wine            eggplant
##         0.0009332089         0.0043994134         0.0131982402
##                eggs           energy bar         energy drink
##         0.1797093721         0.0270630583         0.0266631116
##             escalope extra dark chocolate            flax seed
##         0.0793227570         0.0119984002         0.0090654579
##         french fries          french wine          fresh bread
##         0.1709105453         0.0225303293         0.0430609252
##           fresh tuna        fromage blanc      frozen smoothie
##         0.0222636982         0.0135981869         0.0633248900
##    frozen vegetables      gluten free bar        grated cheese
##         0.0953206239         0.0069324090         0.0523930143
##          green beans         green grapes            green tea
##         0.0086655113         0.0090654579         0.1321157179
##          ground beef                 gums                 ham
##         0.0982535662         0.0134648714         0.0265297960
##     hand protein bar        herb & pepper                honey
##         0.0051993068         0.0494600720         0.0474603386
##             hot dogs              ketchup          light cream
##         0.0323956806         0.0043994134         0.0155979203
##           light mayo       low fat yogurt            magazines
##         0.0271963738         0.0765231302         0.0109318757
##        mashed potato           mayonnaise            meatballs
##         0.0041327823         0.0061325157         0.0209305426
##               melons                 milk        mineral water
##         0.0119984002         0.1295827223         0.2383682176
##                mint       mint green tea              muffins
##         0.0174643381         0.0055992534         0.0241301160
## mushroom cream sauce              napkins          nonfat milk
##         0.0190641248         0.0006665778         0.0103986135
##              oatmeal                  oil            olive oil
##         0.0043994134         0.0230635915         0.0658578856
##             pancakes      parmesan cheese                pasta
##         0.0950539928         0.0198640181         0.0157312358
##               pepper              pet food              pickles
##         0.0265297960         0.0065324623         0.0059992001
##          protein bar              red wine                 rice
##         0.0185308626         0.0281295827         0.0187974937
##                salad               salmon                 salt
##         0.0049326756         0.0425276630         0.0091987735
##             sandwich              shallot              shampoo
##         0.0045327290         0.0077323024         0.0049326756
##               shrimp                 soda                 soup
```

```
##         0.0714571390         0.0062658312         0.0505265965
##            spaghetti      sparkling water              spinach
##         0.1741101187         0.0062658312         0.0070657246
##         strawberries       strong cheese                  tea
##         0.0213304893         0.0077323024         0.0038661512
##         tomato juice         tomato sauce             tomatoes
##         0.0303959472         0.0141314491         0.0683908812
##           toothpaste               turkey       vegetables mix
##         0.0081322490         0.0625249967         0.0257299027
##          water spray           white wine     whole weat flour
##         0.0003999467         0.0165311292         0.0093320891
##    whole wheat pasta     whole wheat rice                 yams
##         0.0294627383         0.0585255299         0.0114651380
##          yogurt cake              zucchini
##         0.0273296894         0.0094654046
```

```
itemFrequencyPlot(supermarket,topN=10,type="absolute",col=brewer.pal(8,'Pastel2'), main="Absolute Item F
```




```
itemFrequencyPlot(supermarket,topN=10,type="relative",col=brewer.pal(8,'Pastel2'),main="Relative Item Fr
```




```
# However since we built the model using 0.006 Min support
# and confidence as 0.25 we obtained 272 rules.
# We use measures of significance and interest on the rules,
# determining which ones are interesting and which to discard.
# However, in order to illustrate the sensitivity of the model to these two parameters,
# we will see what happens if we decrease the support or increase the confidence level
#
association.rules <- apriori(supermarket, parameter = list(supp=0.001, conf=0.8,maxlen=10))
```

```
## 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 [0.00s].
## 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].
```

```
summary(association.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
```

In our first example, we decreased the minimum support of 0.006 to 0.001 and model rules went from 272 to only 74. This would lead us to understand that using a high level of support can make the model lose and using a low confidence level increases the number of rules to quite an extent and many will not be useful.

```
inspect(association.rules[1:10])
```

```
##      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
```

```
shorter.association.rules<- apriori(supermarket, parameter = list(supp=0.001, conf=0.8, maxlen=3))
```

```
## 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
##       3  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 [0.00s].
## checking subsets of size 1 2 3

## Warning in apriori(supermarket, parameter = list(supp = 0.001, conf = 0.8, :
## Mining stopped (maxlen reached). Only patterns up to a length of 3 returned!

##  done [0.01s].
## writing ... [15 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
subset.rules<- which(colSums(is.subset(association.rules, association.rules))>1)
length(subset.rules)
```

```
## [1] 12
```

```
subset.association.rules. <- association.rules[-subset.rules]
```

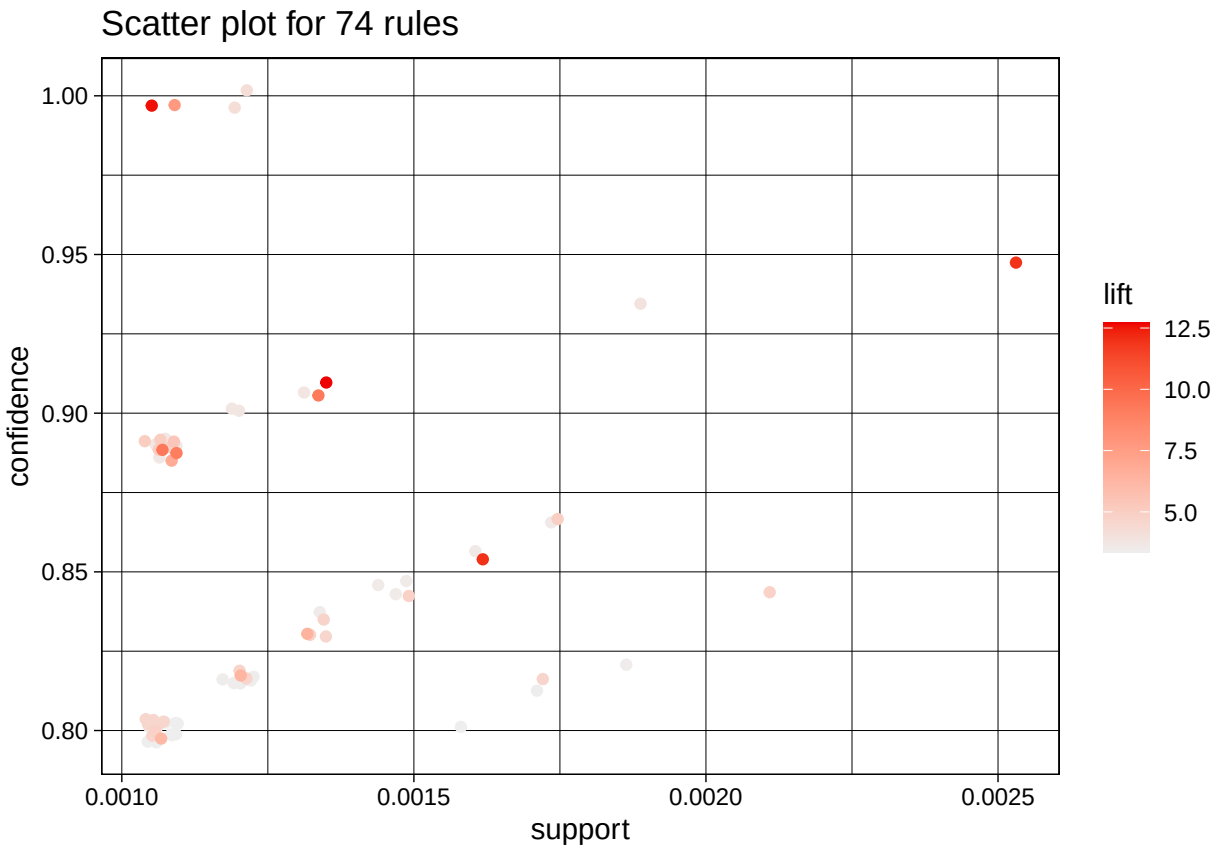
```
bacon.association.rules <- apriori(supermarket, parameter = list(supp=0.001, conf=0.8), appearance = lis
```

```
## 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 ...[1 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 ... [0 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
inspect(head(bacon.association.rules))
```

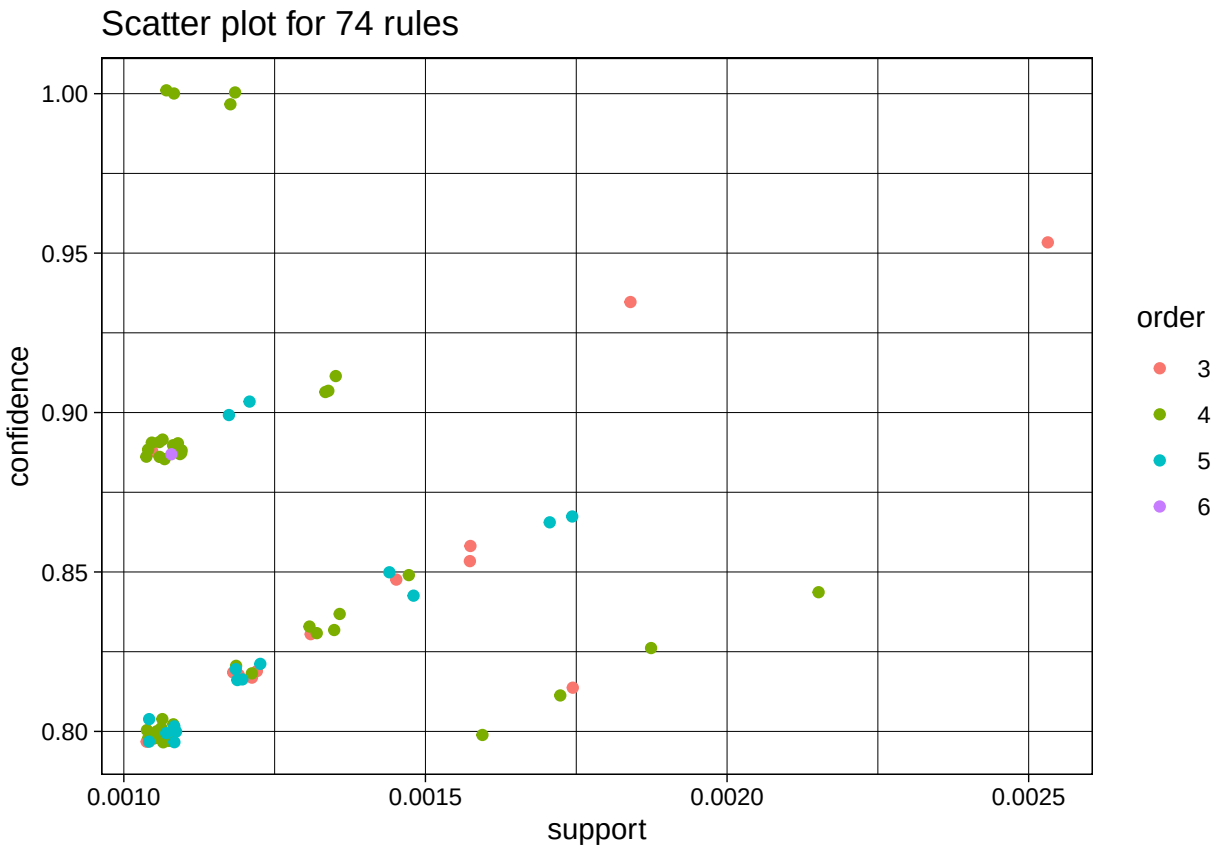
```
library(arulesViz)
subRules <- association.rules[quality(association.rules)$confidence>0.4]
plot(subRules)
```

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```

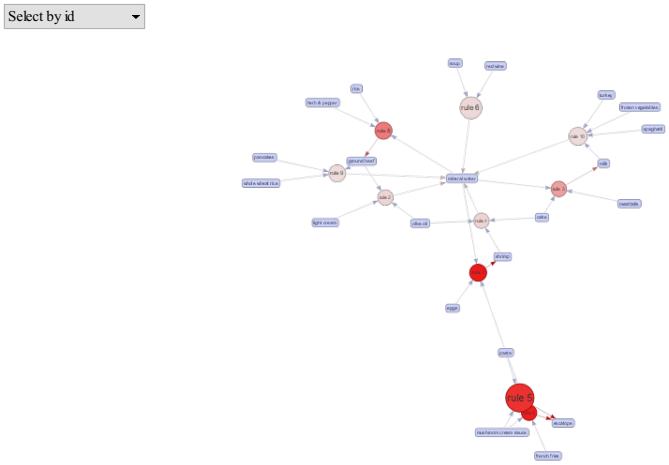



```
plot(subRules, method = "two-key plot")
```

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```




```
top10subRules <- head(subRules, n=10, by = "confidence")
plot(top10subRules, method = "graph", engine= "htmlwidget")
```



```
subRules2<-head(subRules, n=10, by="lift")
plot(subRules2, method = "paracoord")
```