Association Rules Project

Stacy

2022-04-03

```
library(tidyverse)
```

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

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

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

```
library(cluster)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(ggplot2)
library(dendextend)
```

```
##
## ---------------------
## Welcome to dendextend version 1.15.2
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
```

```
##
## Attaching package: 'dendextend'
## The following object is masked from 'package:stats':
##
##     cutree
```

```
library(tidyverse)
library(magrittr)
```

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

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

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

```
library(plotly)
```

```
##
## Attaching package: 'plotly'
```

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

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

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

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(numDeriv)
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
```

```
library (caret)
```

```
## Loading required package: lattice

##
## Attaching package: ’caret’

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

```
library(moments)
library(gridExtra)
```

```
##
## Attaching package: ’gridExtra’

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

```
library(arulesViz)
library(relaimpo)
```

```
## Loading required package: MASS

##
## Attaching package: ’MASS’

## The following object is masked from ’package:plotly’:
##
##     select
## The following object is masked from 'package:dplyr':
##
##     select

## Loading required package: boot

##
## Attaching package: 'boot'

## The following object is masked from 'package:lattice':
##
##     melanoma

## The following object is masked from 'package:psych':
##
##     logit

## Loading required package: survey

## Loading required package: grid

## Loading required package: survival

##
## Attaching package: 'survival'

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

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

##
## Attaching package: 'survey'

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

## Loading required package: mitools

## This is the global version of package relaimpo.

## If you are a non-US user, a version with the interesting additional metric pmvd is available

## from Ulrike Groempings web site at prof.beuth-hochschule.de/groemping.
```

```r
path <-"http://bit.ly/SupermarketDatasetII"

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

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

```r
head(supermarket)
```

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

```r
colnames(supermarket)
```

```
##   [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"
```

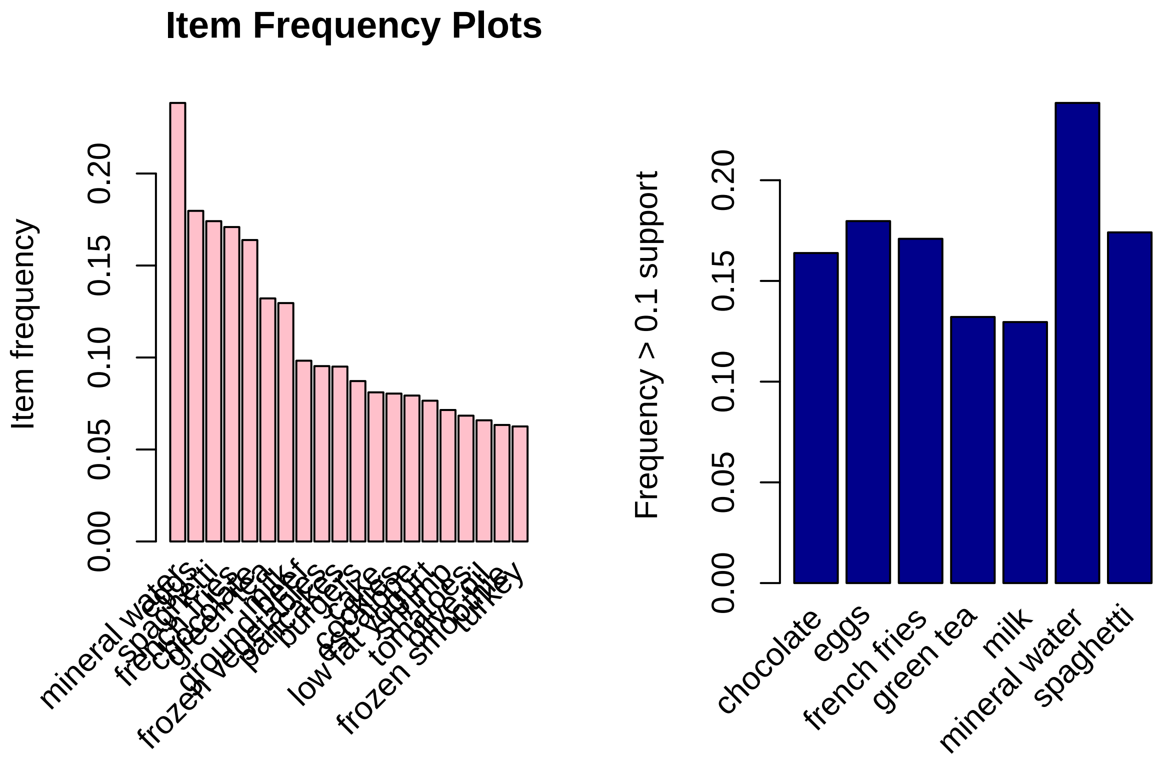
```r
class(supermarket)
```

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

```r
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
```

```r
# Plotting item frequency considering the top 20 items
par(mfcol=c(1,2))
itemFrequencyPlot(supermarket,topN=20,col="pink",
                  ylab="Item frequency",
                  main=" Item Frequency Plots")
itemFrequencyPlot(supermarket,support=0.112,col="darkblue",
                  ylab="Frequency > 0.1 support")
```



```
# Exploring the frequency of some articles
# some operation in percentage terms of the total transactions
itemFrequency(supermarket[, 1:5],type = "absolute")
```

```
##           almonds antioxydant juice         asparagus           avocado
##               153                67                36               250
##       babies food
##                34
```

```
round(itemFrequency(supermarket[, 1:5],type = "relative")*100,2)
```

```
##           almonds antioxydant juice         asparagus           avocado
##              2.04              0.89              0.48              3.33
##       babies food
##              0.45
```

```
# The first rules
rule1<-apriori(supermarket,parameter = list(support=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].
```

```r
rule1
```

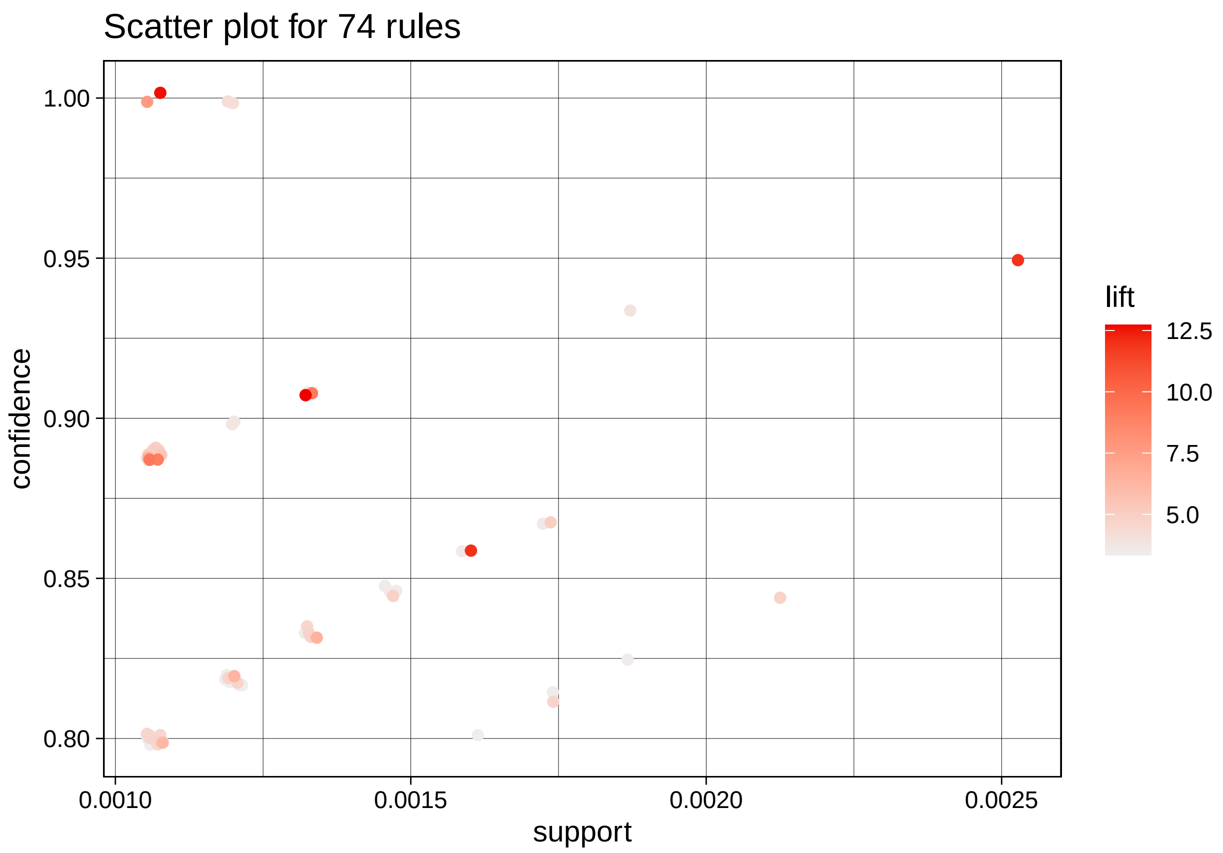
```
## set of 74 rules
```

```r
plot(rule1,type = "graph",control=list(type="items"))
```

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

```
## Available control parameters (with default values):
## main  =  Scatter plot for 74 rules
## colors    =  c("#EE0000FF", "#EEEEEEFF")
## jitter    =  NA
## engine    =  ggplot2
## verbose   =  FALSE
```

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



```
inspect(rule1[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
```

```
sorting1<-sort(rule1,by="confidence",decreasing = TRUE)
inspect(sorting1[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
```

```
# Confidence level for Rule1 is 80%.
#{X} is also called antecedent or left-hand-side (LHS) and
# {Y} is called consequent or right-hand-side (RHS).
#Support is an indication of how frequently a set of items appear in baskets.
# Confidence is an indication of how often the support-rule has been found to be true.
#Lift is a measure of association using both support and confidence.
```

```
# Minimizing support thershold alittle bit
rule2<-apriori(supermarket,parameter =list(support=0.001,conf=0.75))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.75    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 ... [110 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
# Viewing the rules extracted from these
inspect(rule2[1:10])
```

```
##      lhs                              rhs             support     confidence
## [1]  {frozen smoothie, spinach}    => {mineral water} 0.001066524 0.8888889
## [2]  {blueberries, eggs}           => {mineral water} 0.001599787 0.7500000
## [3]  {bacon, pancakes}             => {spaghetti}     0.001733102 0.8125000
## [4]  {nonfat milk, turkey}         => {mineral water} 0.001199840 0.8181818
## [5]  {ground beef, nonfat milk}    => {mineral water} 0.001599787 0.8571429
## [6]  {barbecue sauce, chocolate}   => {mineral water} 0.001333156 0.7692308
## [7]  {mushroom cream sauce, pasta} => {escalope}      0.002532996 0.9500000
## [8]  {milk, pasta}                 => {shrimp}        0.001599787 0.8571429
## [9]  {mineral water, pasta}        => {shrimp}        0.001599787 0.7500000
## [10] {cooking oil, fromage blanc}  => {mineral water} 0.001199840 0.8181818
##      coverage    lift      count
## [1]  0.001199840  3.729058  8
## [2]  0.002133049  3.146393 12
## [3]  0.002133049  4.666587 13
## [4]  0.001466471  3.432428  9
## [5]  0.001866418  3.595877 12
## [6]  0.001733102  3.227069 10
## [7]  0.002666311 11.976387 19
## [8]  0.001866418 11.995203 12
## [9]  0.002133049 10.495802 12
## [10] 0.001466471  3.432428  9
```

```
sorting<-sort(rule2,by="confidence",decreasing = TRUE)
inspect(sorting[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
```

```r
# Getting items that are bought after milk is bought
milk<-subset(rule2,subset=lhs %pin% "milk")
# Sorting items by their confidence level
sorted_milk<-sort(milk,by="confidence",decreasing = TRUE)
# Viewing the top 10 items
inspect(sorted_milk[1:10])
```

```
##      lhs                     rhs                         support confidence    coverage      lift count
## [1]  {frozen vegetables,
##       milk,
##       spaghetti,
##       turkey}             => {mineral water}     0.001199840  0.9000000 0.001333156  3.775671     9
## [2]  {cake,
##       meatballs,
##       milk}               => {mineral water}     0.001066524  0.8888889 0.001199840  3.729058     8
## [3]  {burgers,
##       milk,
##       salmon}             => {spaghetti}         0.001066524  0.8888889 0.001199840  5.105326     8
## [4]  {chocolate,
##       ground beef,
##       milk,
##       mineral water,
##       spaghetti}          => {frozen vegetables} 0.001066524  0.8888889 0.001199840  9.325253     8
## [5]  {ground beef,
##       nonfat milk}        => {mineral water}     0.001599787  0.8571429 0.001866418  3.595877    12
## [6]  {milk,
##       pasta}              => {shrimp}            0.001599787  0.8571429 0.001866418 11.995203    12
## [7]  {frozen vegetables,
##       milk,
##       shrimp,
##       spaghetti}          => {mineral water}     0.001466471  0.8461538 0.001733102  3.549776    11
## [8]  {nonfat milk,
##       turkey}             => {mineral water}     0.001199840  0.8181818 0.001466471  3.432428     9
## [9]  {french fries,
##       herb & pepper,
##       milk}               => {mineral water}     0.001199840   0.8181818 0.001466471   3.432428     9
## [10] {frozen vegetables,
##       milk,
##       olive oil,
##       soup}               => {mineral water}     0.001199840   0.8181818 0.001466471   3.432428     9
```

```
# Getting items purchased before shrimp
shrimp<-subset(rule2,subset=rhs %pin% "shrimp")
# Sorting items by their confidence level
sorted_shrimp<-sort(shrimp,by="confidence",decreasing = TRUE)
# Viewing the top 10 items
inspect(sorted_shrimp[1:3])
```

```
##     lhs                              rhs      support     confidence coverage
## [1] {eggs, mineral water, pasta} => {shrimp} 0.001333156 0.9090909  0.001466471
## [2] {milk, pasta}                => {shrimp} 0.001599787 0.8571429  0.001866418
## [3] {mineral water, pasta}       => {shrimp} 0.001599787 0.7500000  0.002133049
##     lift     count
## [1] 12.72218 10
## [2] 11.99520 12
## [3] 10.49580 12
```

```
# Getting items purchased before escalope
escalope<-subset(rule2,subset=rhs %pin% "escalope")
# Sorting items by their confidence level
sorted_escalope<-sort(escalope,by="confidence",decreasing = TRUE)
# Viewing the top 10 items
inspect(sorted_escalope[1:2])
```

```
##     lhs                           rhs            support confidence    coverage     lift count
## [1] {french fries,
##      mushroom cream sauce,
##      pasta}                 => {escalope} 0.001066524       1.00 0.001066524 12.60672     8
## [2] {mushroom cream sauce,
##      pasta}                 => {escalope} 0.002532996       0.95 0.002666311 11.97639    19
```

```
# Getting items purchased before water
water<-subset(rule2,subset=rhs %pin% "mineral water")
# Sorting items by their confidence level
sorted_water <-sort(water,by="confidence",decreasing = TRUE)
# Viewing the top 10 items
inspect(sorted_water[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
```

```
# Mineral water is the most bought item.
# Support is an indication of how frequently a set of items appear in baskets.
# Confidence is an indication of how often the support-rule has been
# found to be true.
# Lift is a measure of association using both support and confidence.
# The results indicate what is bought alongside the product:
##
```