Association Analysis

Jenipher Mawia

11/13/2020

1. Problem Definition

- create association rules that will allow you to identify relationships between different items bought by customers at Carrefour store.

2. Defining the metrics for success

This project will be considered a success if the following are achieved:

- association rules are generated from the data without any errors

3. 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). The main task here is to generate association rules between different items bought by customers from the store.

4. Experimental Design taken

The project consists of two parts. The following is the order in which I went about to achieve the objectives of this project:

- Data Sourcing and Understanding
- Checking the data
- Data Exploration
- Implementing the solution(generating association rules)
- Visualizing the rules

5. Data Sourcing

The data used was provided by Moringa School and can be downloaded here

Reading the data

```
# load libraries
library(arules)

## Loading required package: Matrix

##
## Attaching package: 'arules'

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

# since we want the data as class transactions, we will read the data using
read.transactions function
association <- read.transactions("http://bit.ly/SupermarketDatasetII")

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

6. Checking the data

```
association

## transactions in sparse format with
##  7501 transactions (rows) and
##  5729 items (columns)
```

There are 7501 transactions and 5729 items in the data

Verifying the object's class

```
# checking the class of our transactions data
class(association)

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

This shows us transactions as the type of data that we will need

7. Data Exploration

```
# Previewing the first 5 transactions
inspect(association[1:5])

##     items
## [1] {cheese,energy,
##      drink,tomato,
##      fat,
##      flour,yams,cottage,
##      grapes,whole,
##      juice,frozen,
##      juice,low,
##      mix,green,
##      oil,
##      shrimp,almonds,avocado,vegetables,
##      smoothie,spinach,olive,
##      tea,honey,salad,mineral,
##      water,salmon,antioxydant,
##      weat,
##      yogurt,green}
## [2] {burgers,meatballs,eggs}
## [3] {chutney}
## [4] {turkey,avocado}
## [5] {bar,whole,
##      mineral,
##      rice,green,
##      tea,
##      water,milk,energy,
##      wheat}
```

The transactions vary from one item to a group of more than one item.

```
# preview the items that make up our dataset,
items<-as.data.frame(itemLabels(association))
colnames(items) <- "Item"
head(items,15)
```

```
##                                       Item
## 1                                        &
## 2                              accessories
## 3                  accessories,antioxydant
## 4              accessories,champagne,fresh
## 5            accessories,champagne,protein
## 6                    accessories,chocolate
## 7  accessories,chocolate,champagne,frozen
## 8             accessories,chocolate,frozen
## 9                accessories,chocolate,low
## 10      accessories,chocolate,pasta,salt
## 11      accessories,chocolate,salt,green
## 12                     accessories,cookies
## 13                     accessories,cottage
## 14                    accessories,escalope
## 15                      accessories,french
```

```
# 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(association)
```

```
## transactions as itemMatrix in sparse format with
##  7501 rows (elements/itemsets/transactions) and
##  5729 columns (items) and a density of 0.0005421748
##
## most frequent items:
##     tea   wheat mineral     fat  yogurt (Other)
##     803     645     577     574     543   20157
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   15   16
## 1603 2007 1382  942  651  407  228  151   70   39   13    5    1    1    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   3.106   4.000  16.000
##
## includes extended item information - examples:
##                    labels
## 1                       &
## 2             accessories
## 3 accessories,antioxydant
```

The most frequent items are: tea, wheat, mineral, fat, yogurt

Element (itemset/transaction) length distribution: This gives us how many transactions are there for 1-itemset, for 2-itemset and so on.

For example, there are 1603 transactions for one item, 2007 transactions for 2 items, and there are 16 items in one transaction which is the longest/most items purchased in one transaction.

Item Frequency

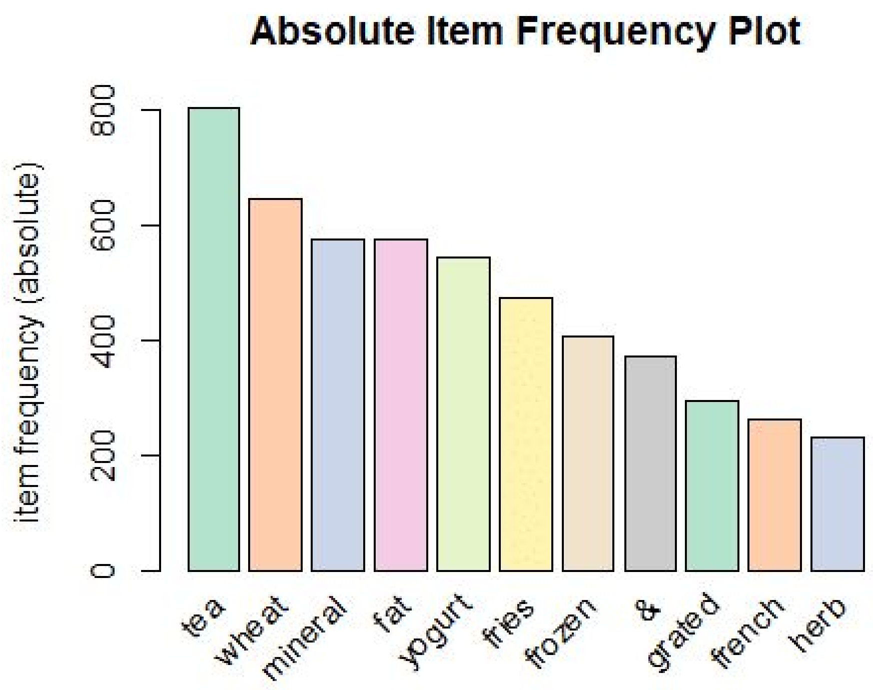
```
# Exploring the frequency of some articles i.e. transactions ranging from 12
to 16
itemFrequency(association[, 12:16],type = "absolute")
```

```
##  accessories,cookies  accessories,cottage accessories,escalope
##                    5                    2                    1
##   accessories,french    accessories,fresh
##                   13                    2
```

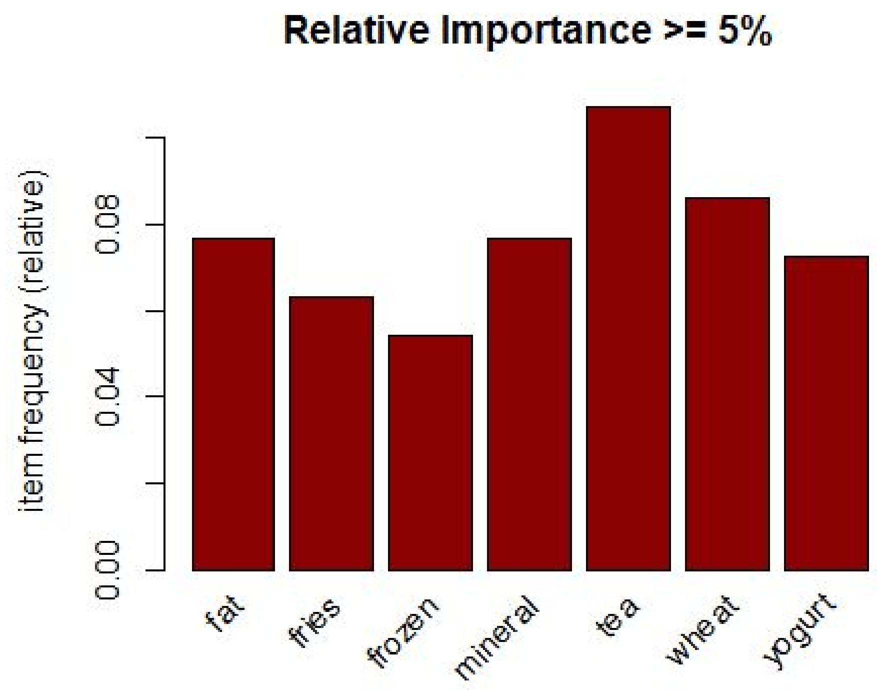
Graphical Analysis of Item frequency

```
# Producing a chart of frequencies and fitering

par(mfrow = c(1, 1))

# Create an item frequency plot for the top 10 most common items
if (!require("RColorBrewer")) {
  # install color package of R
install.packages("RColorBrewer")
```

```r
#include library RColorBrewer
library(RColorBrewer)
}

## Loading required package: RColorBrewer

itemFrequencyPlot(association,topN=11,type="absolute",col=brewer.pal(8,'Paste
l2'), main="Absolute Item Frequency Plot")
```



```r
# and the items whose relative importance is at least 5%

itemFrequencyPlot(association, support = 0.05,col="darkred", main="Relative
Importance >= 5%")
```



This plot shows that ‘Tea’ and ‘Wheat’ have the most sales. So to increase the sale of ‘herb’ the retailer can put it near ‘Tea’.

8. Implementing the solution

Building an Apriori model to generate association rules

```
# 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 (association, 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 ...[5729 item(s), 7501 transaction(s)] done [0.02s].
## sorting and recoding items ... [354 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [271 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
# checking the rules
rules
```

```
## set of 271 rules
```

Since there are 271 rules, we print only top 10:

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

```
##      lhs                                rhs      support     confidence
## [1]  {cookies,low}                   => {yogurt} 0.001066524 1.0
## [2]  {cookies,low}                   => {fat}    0.001066524 1.0
## [3]  {extra}                         => {dark}   0.001066524 1.0
## [4]  {burgers,whole}                 => {wheat}  0.001199840 1.0
## [5]  {fries,escalope,pasta,mushroom} => {cream}  0.001066524 1.0
## [6]  {fries,cookies,green}           => {tea}    0.001333156 1.0
## [7]  {shrimp,whole}                  => {wheat}  0.001066524 1.0
## [8]  {rice,cake}                     => {wheat}  0.001333156 1.0
## [9]  {tomatoes,whole}                => {wheat}  0.001066524 0.8
## [10] {rice,chocolate}                => {wheat}  0.001199840 0.9
##      coverage    lift      count
## [1]  0.001066524 13.813996  8
## [2]  0.001066524 13.067944  8
## [3]  0.001066524 83.344444  8
## [4]  0.001199840 11.629457  9
## [5]  0.001066524 47.777070  8
## [6]  0.001333156  9.341220 10
## [7]  0.001066524 11.629457  8
## [8]  0.001333156 11.629457 10
## [9]  0.001333156  9.303566  8
## [10] 0.001333156 10.466512  9
```

This would tell us the items that the customers bought before purchasing other items. For example:

- From the confidence levels, 100% of customers who bought “cookies and low” also bought “fat” or “yogurt”
- 100% of customers who bought “burgers and whole” also bought “wheat”

```
# check summary of the rules
summary(rules)

## set of 271 rules
##
## rule length distribution (lhs + rhs):sizes
##   2   3   4
## 107 144  20
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   3.000   2.679   3.000   4.000
##
## summary of quality measures:
##     support           confidence       coverage             lift
##  Min.   :0.001067   Min.   :0.800   Min.   :0.001067   Min.   :  7.611
##  1st Qu.:0.001200   1st Qu.:0.931   1st Qu.:0.001200   1st Qu.: 11.630
##  Median :0.001600   Median :1.000   Median :0.001600   Median : 13.068
##  Mean   :0.002834   Mean   :0.963   Mean   :0.002973   Mean   : 22.372
##  3rd Qu.:0.002666   3rd Qu.:1.000   3rd Qu.:0.002800   3rd Qu.: 20.218
##  Max.   :0.068391   Max.   :1.000   Max.   :0.076523   Max.   :613.718
##      count
##  Min.   :  8.00
##  1st Qu.:  9.00
##  Median : 12.00
##  Mean   : 21.26
##  3rd Qu.: 20.00
##  Max.   :513.00
##
## mining info:
##         data ntransactions support confidence
##  association          7501   0.001        0.8
```

The summary shows:

- the total number of rules: 271 rules
- Distribution of rule length: A length of 3 items has the most rules: 144 and length of 4 items have the lowest number of rules:20
- Summary of Quality measures: Min and max values for Support, Confidence and, Lift.
- Information used for creating rules: The data, support, and confidence we provided to the algorithm.

Limiting the number and size of rules

We use measures of significance and interest on the rules, determining which ones are interesting and which to discard.

However since we built the model using 0.001 Min support and confidence as 0.8 we obtained 271 rules.To illustrate the sensitivity of the model to these two parameters, we will see what happens if we increase the support or lower the confidence level.

For stronger rules, you can increase the value of conf and for more extended rules give higher value to maxlen or adjust the supp parameter.

```
# Building a apriori model with Min Support as 0.002 and confidence as 0.8.
rules2 <- apriori (association,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 ...[5729 item(s), 7501 transaction(s)] done [0.05s].
## sorting and recoding items ... [189 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [99 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

# checking the rules
rules2

## set of 99 rules
```

We get 99 rules with supp=0.02 and conf=0.8. This would lead us to understand that using a high level of support can make the model lose interesting rules.

```
# Building apriori model with Min Support as 0.002 and confidence as 0.6.
rules3 <- apriori (association, 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 ...[5729 item(s), 7501 transaction(s)] done [0.03s].
## sorting and recoding items ... [354 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [319 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules3
```

```
## set of 319 rules
```

We get 319 rules with parameters: supp = 0.001, conf = 0.6. This would mean that using a low confidence level increases the number of rules to quite an extent and many will not be useful.

Removing redundant rules

Here, we reduce the number of rules by removing rules that are subsets of larger rules.

```
# get subset rules in vector
subset.rules <- which(colSums(is.subset(rules, rules)) > 1)
# number of subset rules
length(subset.rules)
```

```
## [1] 163
```

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

```
## set of 108 rules
```

We now have a set of 108 rules which we can make better sense of as they are not many.

9. Finding Rules related to given items

```
# If we're interested in making a promotion relating to the sale of yogurt,
# we could create a subset of rules concerning these products
# ---
# This would tell us the items that the customers bought before purchasing
yogurt
# ---
#
```

```r
yogurt_rules <- apriori(association, parameter = list(supp=0.001, conf=0.8),
appearance = list(default="lhs", rhs="yogurt"))

## 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 ...[5729 item(s), 7501 transaction(s)] done [0.04s].
## sorting and recoding items ... [354 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [58 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

# check the rules
inspect(yogurt_rules[1:10])

##      lhs               rhs      support     confidence coverage    lift
## [1]  {cookies,low} => {yogurt} 0.001066524 1.0000000  0.001066524 13.81400
## [2]  {cake,low}    => {yogurt} 0.001066524 0.8888889  0.001199840 12.27911
## [3]  {water,low}   => {yogurt} 0.001199840 0.9000000  0.001333156 12.43260
## [4]  {wine,low}    => {yogurt} 0.001333156 1.0000000  0.001333156 13.81400
## [5]  {sauce,low}   => {yogurt} 0.001199840 0.9000000  0.001333156 12.43260
## [6]  {dogs,low}    => {yogurt} 0.001066524 0.8000000  0.001333156 11.05120
## [7]  {cheese,low}  => {yogurt} 0.001733102 1.0000000  0.001733102 13.81400
## [8]  {mayo,low}    => {yogurt} 0.001733102 1.0000000  0.001733102 13.81400
## [9]  {bar,low}     => {yogurt} 0.001599787 0.8000000  0.001999733 11.05120
## [10] {oil,low}     => {yogurt} 0.002399680 0.8571429  0.002799627 11.84057
##      count
## [1]   8
## [2]   8
## [3]   9
## [4]  10
## [5]   9
## [6]   8
## [7]  13
## [8]  13
## [9]  12
## [10] 18
```

We can conclude that most customers bought “low” before buying “yogurt”. The marketing team can put these two products next to each other.

```
# Which items did the customers buy before purchasing tea

tea_rules <- apriori(association, parameter = list(supp=0.001, conf=0.8),
appearance = list(default="lhs", rhs="tea"))
```

```
## 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 ...[5729 item(s), 7501 transaction(s)] done [0.06s].
## sorting and recoding items ... [354 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.01s].
## writing ... [14 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
#check the rukes
inspect(tea_rules[1:10])
```

```
##      lhs                         rhs   support     confidence coverage
lift
## [1]  {fries,cookies,green} => {tea} 0.001333156 1.0000000  0.001333156
9.341220
## [2]  {smoothie,green}      => {tea} 0.002133049 1.0000000  0.002133049
9.341220
## [3]  {mayo,green}          => {tea} 0.002133049 1.0000000  0.002133049
9.341220
## [4]  {drink,green}         => {tea} 0.002133049 1.0000000  0.002133049
9.341220
## [5]  {bar,green}           => {tea} 0.001733102 0.8666667  0.001999733
8.095724
## [6]  {cake,green}          => {tea} 0.002133049 0.9411765  0.002266364
8.791737
## [7]  {dogs,green}          => {tea} 0.002133049 0.9411765  0.002266364
8.791737
## [8]  {cheese,green}        => {tea} 0.002666311 0.9090909  0.002932942
8.492019
```

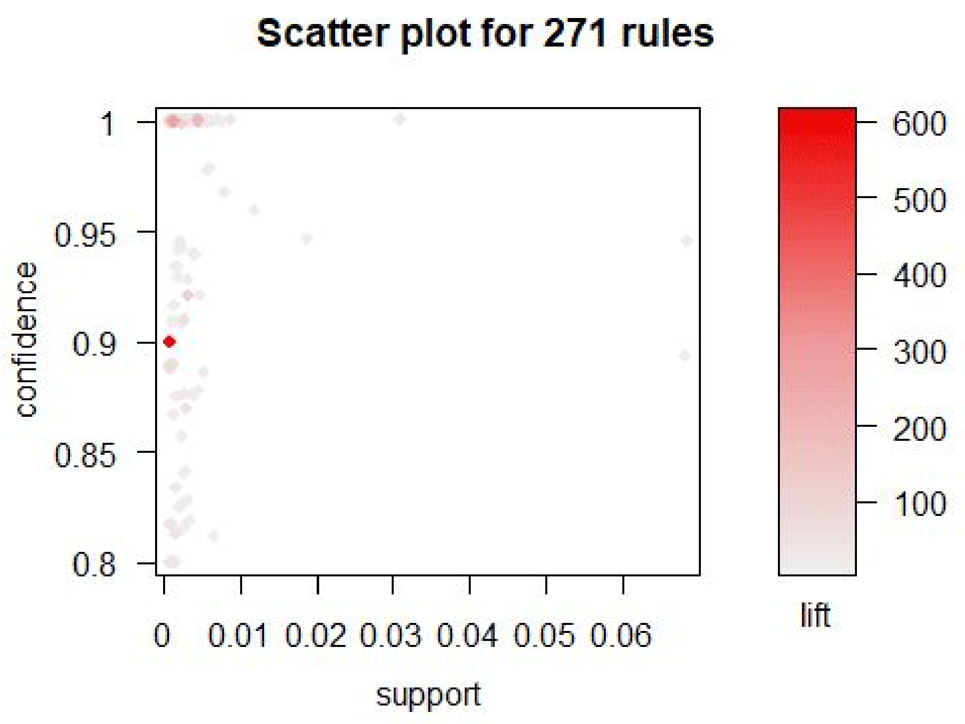
```
## [9]  {juice,green}         => {tea} 0.002932942 0.8148148  0.003599520
7.611365
## [10] {bread,green}         => {tea} 0.004132782 0.9393939  0.004399413
8.775086
##      count
## [1]  10
## [2]  16
## [3]  16
## [4]  16
## [5]  13
## [6]  16
## [7]  16
## [8]  20
## [9]  22
## [10] 31
```

Most customers bought “Green” before buying “tea”.

10. Visualizing Association Rules

Scatter plot

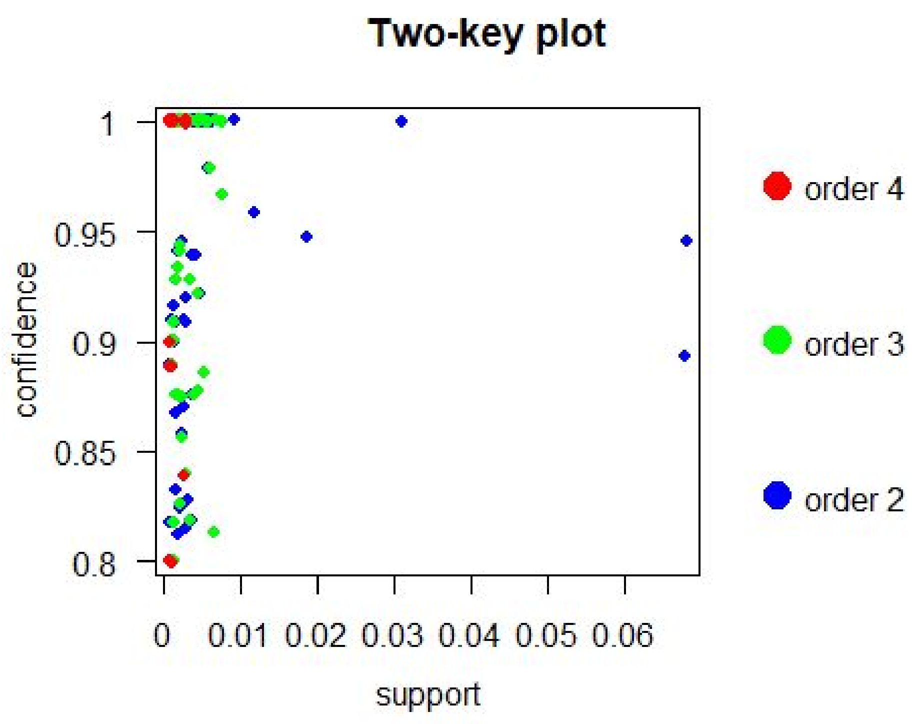
```
library(arulesViz)

## Loading required package: grid

# Filter rules with confidence greater than 0.4 or 40%
subRules<-rules[quality(rules)$support>0.001]
#Plot SubRules
plot(subRules)

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



The above plot shows that rules with high lift have high confidence.

Two Key Plot

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

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



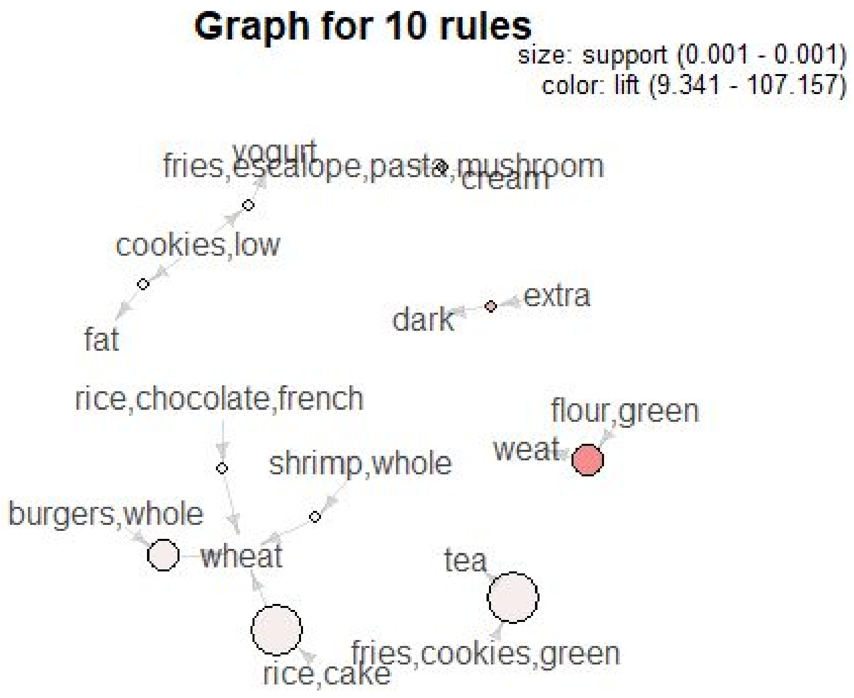
The two-key plot uses support and confidence on x and y-axis respectively. It uses order for coloring. The order is the number of items in the rule. Order 2 has higher values for the support compared to order 3 and 4.

Interactive Scatter-Plot

```
# plotly_arules(subRules)
```

Graph based visualizations

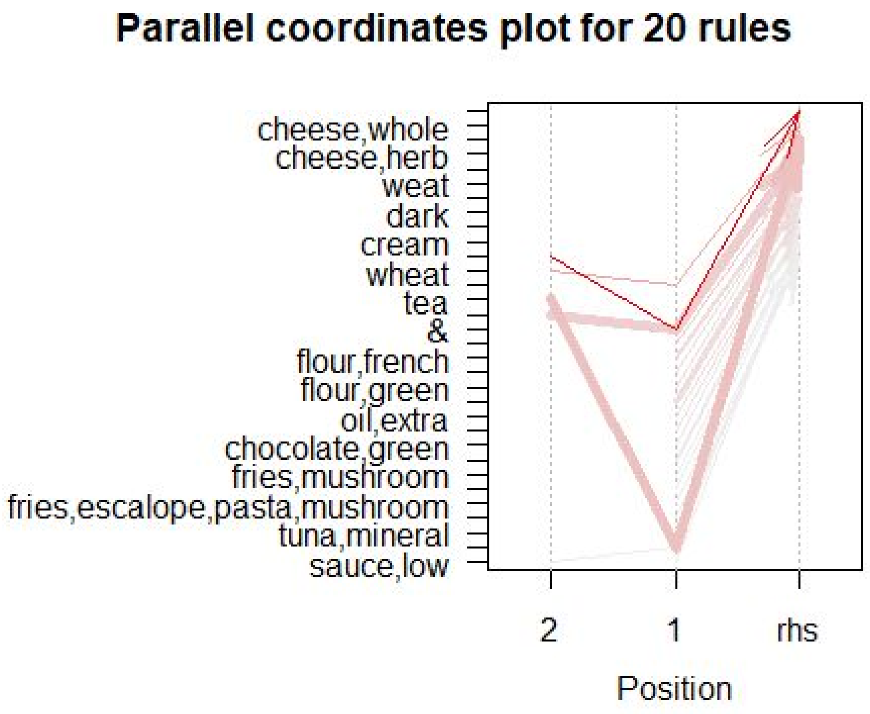
```
top10subRules <- head(subRules, n = 10, by = "confidence")

plot(top10subRules, method = "graph") #,engine = "htmlwidget")
```



- We can see that for rule 1 and 2, customers who bought “cookies and low” also bought “yogurt” or “fat”
- For rule 3, customers who bought “dark” also bought “extra”
- For rule 5, customers who bought “fries, escalope, pasta, mushroom” also bought “cream” etc.

Individual Rule Representation

```
# Filter top 20 rules with highest lift
subRules2<-head(subRules, n=20, by="lift")
plot(subRules2, method="paracoord")
```



The RHS is the Consequent or the item we propose the customer will buy; the positions are in the LHS where 2 is the most recent addition to our basket and 1 is the item we previously had.

Look at the topmost arrow. It shows that when a customer has “cream” and “tea” in their shopping cart, they are more likely to buy “cheese and whole”