Market Basket Analyses

Gizem Guleli

2023-02-04

```
library (arules)
library (dplyr)
library(plyr)
library(tidyr)
library(tidyverse)
library (arulesViz)
library (Rcpp)
```

DATA

The Groceries dataset is a collection of 38765 transactions made at a grocery store. Each transaction contains a list of items purchased by a customer. This dataset can be used for market basket analysis, which is a technique used to identify patterns in customer purchasing behavior.

I first read data set and then clean missing values and dublicates row.

```
#read and summarize data
groceries <- read.csv("Groceries_dataset.csv")

head(groceries)
```

```
##   Member_number       Date  itemDescription
## 1          1808 21-07-2015   tropical fruit
## 2          2552 05-01-2015       whole milk
## 3          2300 19-09-2015        pip fruit
## 4          1187 12-12-2015 other vegetables
## 5          3037 01-02-2015       whole milk
## 6          4941 14-02-2015       rolls/buns
```

```
summary(groceries)
```

```
##  Member_number      Date           itemDescription
##  Min.   :1000   Length:38765       Length:38765
##  1st Qu.:2002   Class :character   Class :character
##  Median :3005   Mode  :character   Mode  :character
##  Mean   :3004
##  3rd Qu.:4007
##  Max.   :5000
```

```
# Remove any duplicates
groceries <- distinct(groceries)
```

```r
# Convert the Date column to a date format
groceries$Date <- as.Date(groceries$Date, format = "%Y-%m-%d")

# Remove any rows with missing values
groceries2 <- drop_na(groceries)

###Change the name of variables to make more easy

colnames(groceries2)[colnames(groceries2)=="itemDescription"] <- "Item"

###Check last version of dataset
head(groceries2)
```

```
##   Member_number       Date             Item
## 1          1808 0021-07-20   tropical fruit
## 2          2552 0005-01-20       whole milk
## 3          2300 0019-09-20        pip fruit
## 4          1187 0012-12-20 other vegetables
## 5          3037 0001-02-20       whole milk
## 6          4941 0014-02-20       rolls/buns
```

```r
dim(groceries2)
```

```
## [1] 38006     3
```

```r
str(groceries2)
```

```
## 'data.frame':    38006 obs. of  3 variables:
##  $ Member_number: int  1808 2552 2300 1187 3037 4941 4501 3803 2762 4119
...
##  $ Date         : Date, format: "0021-07-20" "0005-01-20" ...
##  $ Item         : chr  "tropical fruit" "whole milk" "pip fruit" "other
vegetables" ...
```

```r
###Order data according to member_number
sorted <- groceries2[order(groceries2$Member_number),]


###Group all the items that  bought  by the same customer on the same date to
see set of items
itemList <- aggregate(Item ~ Member_number + Date, data = groceries2, FUN =
function(x) paste(x, collapse = ","))

head(itemList,5)
```

```
##   Member_number       Date                                Item
## 1          1220 0001-01-20    canned beer,margarine,chocolate
## 2          1235 0001-01-20 sausage,bottled beer,spread cheese
## 3          1249 0001-01-20                citrus fruit,coffee
## 4          1381 0001-01-20                           curd,soda
## 5          1422 0001-01-20  tropical fruit,turkey,salty snack
```

```
dim(itemList)

## [1] 14934     3
```

ANALYSE

First of all the original groceries data frame is subsetted to select only the column containing the items , and then converted the resulting data frame of items to a transaction object using read.transaction() function because the transaction object is the preferred format for performing market basket analysis using the arules package in R.

By converting the item data into a transaction object, we can use the functions in the arules package to perform association rule mining, which involves finding patterns in the data such as frequent itemsets (groups of items that are frequently purchased together) and association rules. In the end I print the total number of transactions and items in the dataset.As we see from the result there are 14935 transaction and 168 items.

```
subset <- itemList[,3]
write.csv(subset,"subset", quote = FALSE, row.names = TRUE)
head(subset)

## [1] "canned beer,margarine,chocolate"    "sausage,bottled beer,spread
cheese"
## [3] "citrus fruit,coffee"                "curd,soda"
## [5] "tropical fruit,turkey,salty snack"  "other vegetables,yogurt"

dim(subset)

## NULL

trans1 = read.transactions(file="subset", rm.duplicates= TRUE,
format="basket",sep=",",cols=1);

## distribution of transactions with duplicates:
## 1
## 3

head(trans1)

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

dim(trans1)

## [1] 14935   168

length(trans1)

## [1] 14935

LIST(head(trans1))
```

```
## [[1]]
## [1] "x"
##
## $`1`
## [1] "canned beer" "chocolate"   "margarine"
##
## $`2`
## [1] "bottled beer"  "sausage"       "spread cheese"
##
## $`3`
## [1] "citrus fruit" "coffee"
##
## $`4`
## [1] "curd" "soda"
##
## $`5`
## [1] "salty snack"    "tropical fruit" "turkey"
```

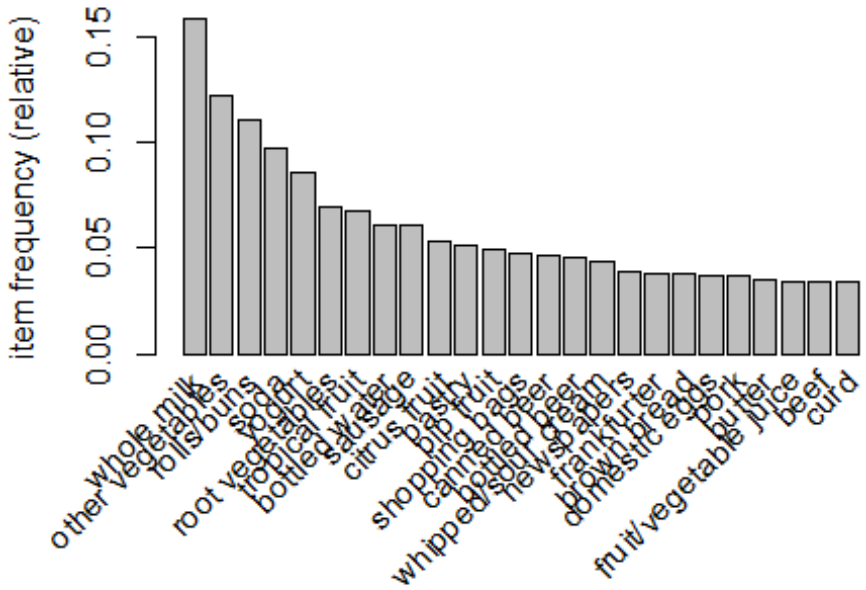
```
# Print the number of transactions and items in the dataset
cat("Number of transactions:", length(trans1), "\n")
```

```
## Number of transactions: 14935
```

```
cat("Number of items:", length(itemLabels(trans1)), "\n")
```

```
## Number of items: 168
```

```
itemFrequencyPlot(trans1, topN = 25)
```



Calculates the relative and absolute frequency of occurrence of each item in the transaction dataset trans1. First frequencies are expressed as a proportion between 0 and 1, where 1 represents an item that appears in every transaction and 0 represents an item that does not appear in any transaction while the second (absolute) frequencies are expressed as infinite integers that represent the number of transactions in which the item appears. To see the most occured items I ordered result as descending and results show that whole milk, other vegetables and rollss/buns are the most occured 3 items.

```
head(sort(round(itemFrequency(trans1),3),decreasing = TRUE))
```

```
##       whole milk other vegetables       rolls/buns             soda
##            0.158            0.122            0.110            0.097
##           yogurt  root vegetables
##            0.086            0.070
```

```
head(sort(itemFrequency(trans1, type="absolute"),decreasing = TRUE))
```

```
##       whole milk other vegetables       rolls/buns             soda
##             2363             1827             1646             1453
##           yogurt  root vegetables
##             1285             1041
```

Algorithms

Eclat

```
sets<-eclat(trans1, parameter = list( sup = 0.001 , maxlen=15))
```

```
## Eclat
##
## parameter specification:
##  tidLists support minlen maxlen            target  ext
##     FALSE   0.001      1     15 frequent itemsets TRUE
##
## algorithmic control:
##  sparse sort verbose
##       7   -2    TRUE
##
## Absolute minimum support count: 14
##
## create itemset ...
## set transactions ...[168 item(s), 14935 transaction(s)] done [0.00s].
## sorting and recoding items ... [149 item(s)] done [0.00s].
## creating sparse bit matrix ... [149 row(s), 14935 column(s)] done [0.00s].
## writing  ... [753 set(s)] done [0.01s].
## Creating S4 object  ... done [0.00s].
```

```
class(sets)
```

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

```
inspect(head(sets, n = 5))
```

```
##     items                             support     count
## [1] {frozen fish, whole milk}         0.001071309 16
## [2] {rolls/buns, seasonal products}   0.001004352 15
## [3] {pot plants, whole milk}          0.001004352 15
## [4] {other vegetables, pot plants}    0.001004352 15
## [5] {pasta, whole milk}               0.001071309 16
```

```
head(round(support(items(sets), trans1) , 2))
```

```
## [1] 0 0 0 0 0 0
```

```
# getting rules
sets<-ruleInduction(sets, trans1, confidence=0.08)
```

```
## Warning: as(<dgCMatrix>, "ngCMatrix") is deprecated since Matrix 1.5-0; do
as(.,
## "nMatrix") instead
```

```
# screening the  part of the rules
inspect(head(sets))
```

```
##     lhs                     rhs                support     confidence lift
## [1] {frozen fish}        => {whole milk}       0.001071309 0.1568627
0.9914283
## [2] {seasonal products}  => {rolls/buns}       0.001004352 0.1415094
1.2839875
## [3] {pot plants}         => {whole milk}       0.001004352 0.1282051
0.8103020
## [4] {pot plants}         => {other vegetables} 0.001004352 0.1282051
1.0480260
## [5] {pasta}              => {whole milk}       0.001071309 0.1322314
0.8357495
## [6] {pickled vegetables} => {whole milk}       0.001004352 0.1119403
0.7075025
##     itemset
## [1] 1
## [2] 2
## [3] 3
## [4] 4
## [5] 5
## [6] 6
```

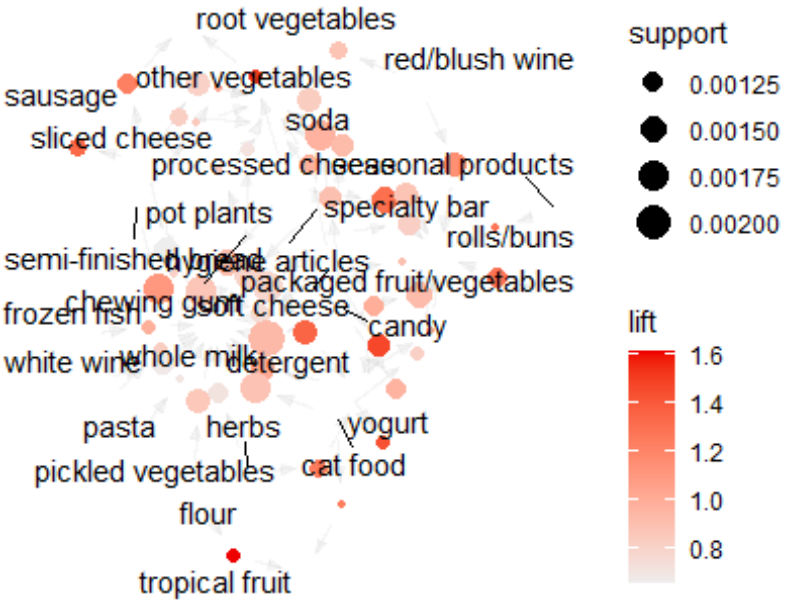
```
summary(sets)
```

```
## set of 234 rules
##
## rule length distribution (lhs + rhs):sizes
##   2   3
## 209  25
##
```

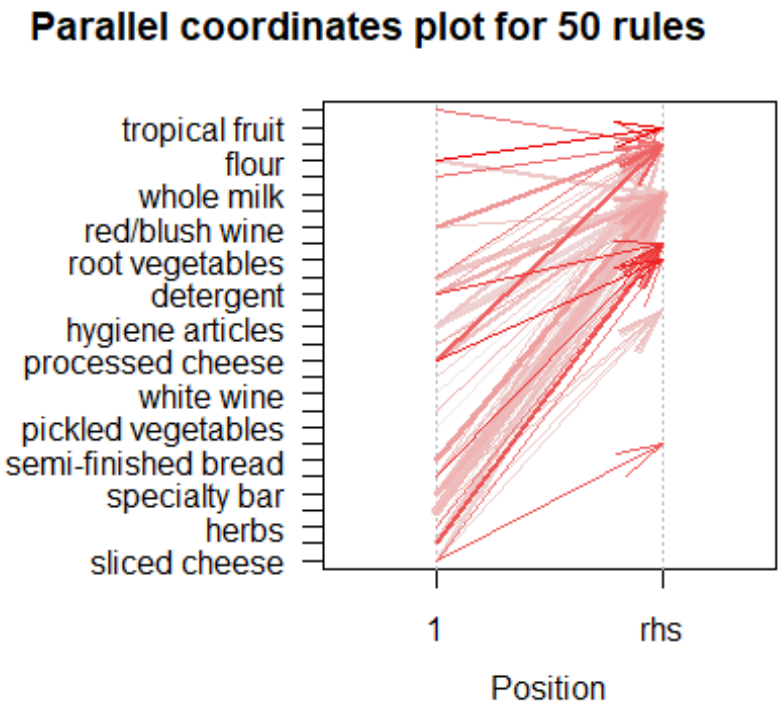
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   2.000   2.107   2.000   3.000
##
## summary of quality measures:
##     support           confidence          lift            itemset
##  Min.   :0.001004   Min.   :0.08038   Min.   :0.5289   Min.   :  1.00
##  1st Qu.:0.001339   1st Qu.:0.08922   1st Qu.:0.7787   1st Qu.: 66.25
##  Median :0.001975   Median :0.10411   Median :0.8480   Median :223.50
##  Mean   :0.003056   Mean   :0.10954   Mean   :0.9148   Mean   :281.18
##  3rd Qu.:0.003917   3rd Qu.:0.12640   3rd Qu.:0.9803   3rd Qu.:528.00
##  Max.   :0.014931   Max.   :0.25581   Max.   :2.1659   Max.   :604.00
##
## mining info:
##    data ntransactions support
##  trans1         14935   0.001
##                                                                  call
confidence
##  eclat(data = trans1, parameter = list(sup = 0.001, maxlen = 15))
0.08
```

```
plot(sets, jitter = 0)
```


```
plot(sets[1:50], method="graph")
```




```
plot(sets[1:50], method="paracoord")
```

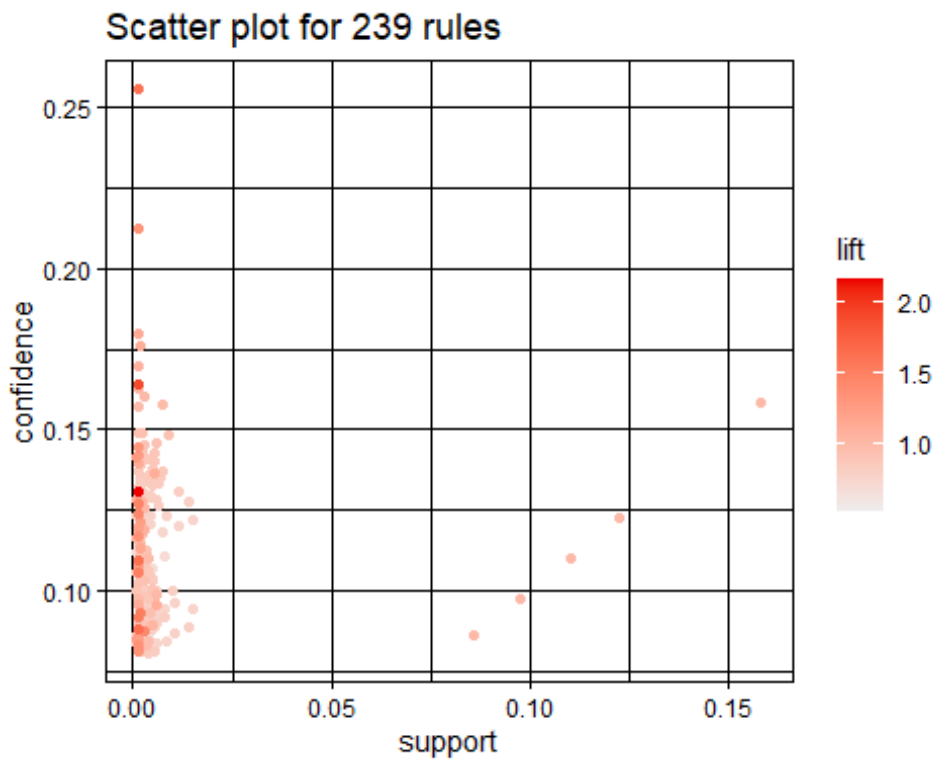


The Apriori

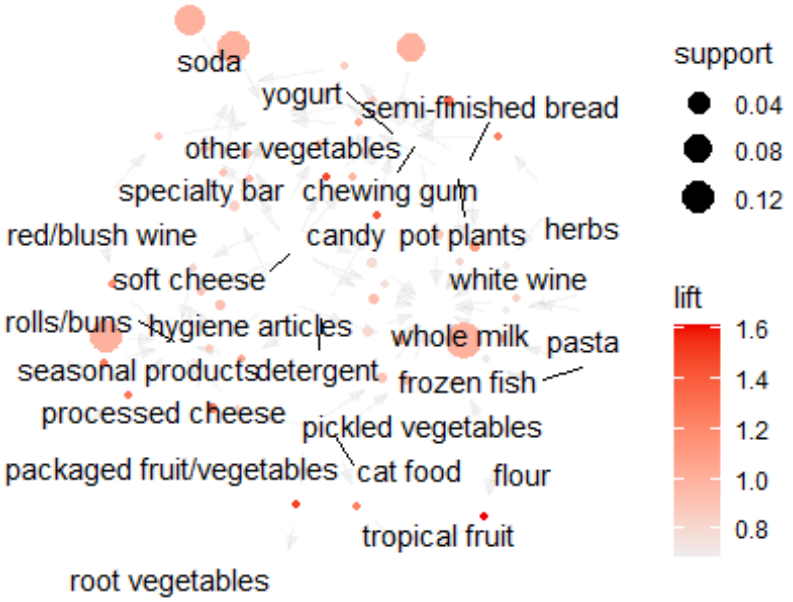
```
# creating rules - standard settings
rules.trans1<-apriori(trans1, parameter=list(supp=0.001 , conf=0.08))

## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.08    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: 14
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[168 item(s), 14935 transaction(s)] done [0.00s].
## sorting and recoding items ... [149 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [239 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

summary(rules.trans1)

## set of 239 rules
##
## rule length distribution (lhs + rhs):sizes
##   1   2   3
##   5 209  25
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   2.084   2.000   3.000
##
## summary of quality measures:
##     support           confidence         coverage             lift
##  Min.   :0.001004   Min.   :0.08038   Min.   :0.005357   Min.   :0.5289
##  1st Qu.:0.001339   1st Qu.:0.08925   1st Qu.:0.013994   1st Qu.:0.7799
##  Median :0.002143   Median :0.10429   Median :0.018815   Median :0.8496
##  Mean   :0.005394   Mean   :0.10965   Mean   :0.049330   Mean   :0.9166
##  3rd Qu.:0.004017   3rd Qu.:0.12619   3rd Qu.:0.037697   3rd Qu.:0.9944
##  Max.   :0.158219   Max.   :0.25581   Max.   :1.000000   Max.   :2.1659
##       count
##  Min.   :  15.00
##  1st Qu.:  20.00
##  Median :  32.00
##  Mean   :  80.56
```

```
##  3rd Qu.:  60.00
##  Max.   :2363.00
##
## mining info:
##    data ntransactions support confidence
##  trans1         14935   0.001       0.08
##                                                                  call
##  apriori(data = trans1, parameter = list(supp = 0.001, conf = 0.08))
```

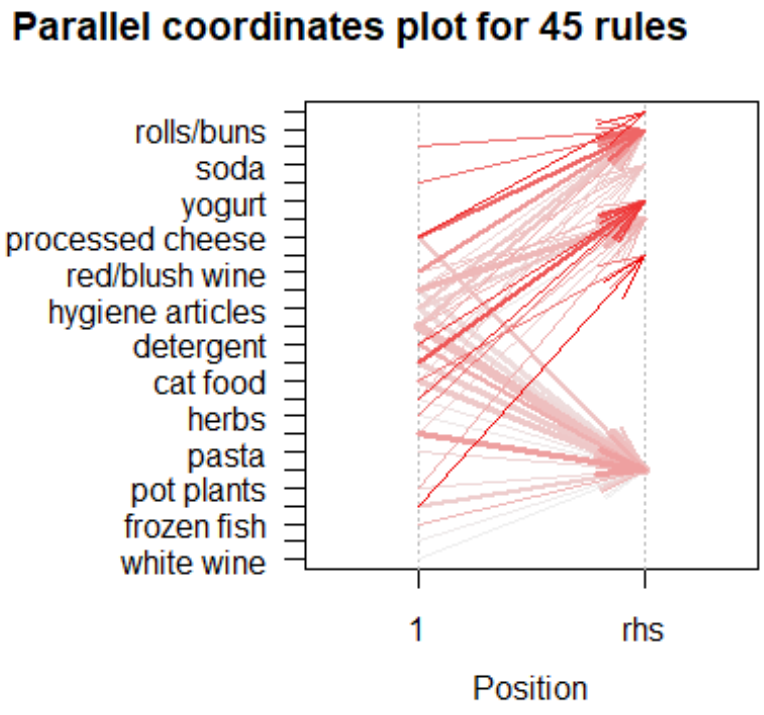
```
plot(rules.trans1, jitter = 0)
```




```
plot(rules.trans1[1:50], method="graph")
```




```
plot(rules.trans1[1:50], method="paracoord")
```



```r
# Changing rules - standard settings
rules.trans2<-apriori(trans1, parameter=list(supp=0.002 , conf=0.05))
```

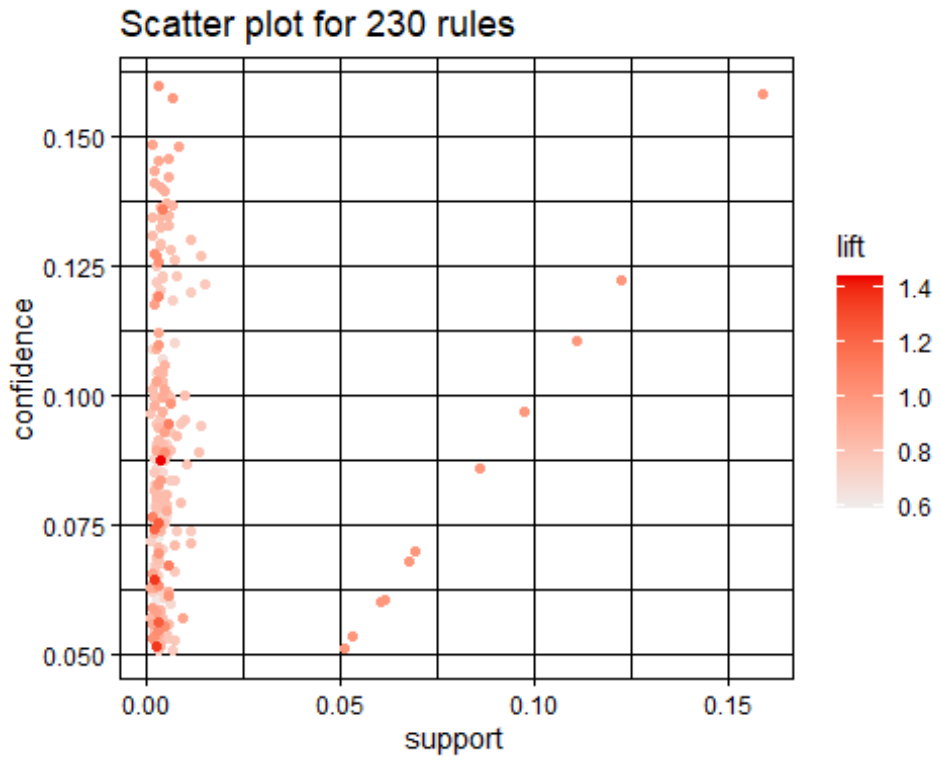
```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.05    0.1    1 none FALSE            TRUE       5   0.002      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 29
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[168 item(s), 14935 transaction(s)] done [0.00s].
## sorting and recoding items ... [126 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [230 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
summary(rules.trans2)
```

```
## set of 230 rules
##
## rule length distribution (lhs + rhs):sizes
##   1   2
##  11 219
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   1.952   2.000   2.000
##
## summary of quality measures:
##     support           confidence         coverage            lift
##  Min.   :0.002009   Min.   :0.05094   Min.   :0.01440   Min.   :0.5914
##  1st Qu.:0.002561   1st Qu.:0.06394   1st Qu.:0.03401   1st Qu.:0.7701
##  Median :0.003482   Median :0.08159   Median :0.04540   Median :0.8233
##  Mean   :0.008135   Mean   :0.08733   Mean   :0.09672   Mean   :0.8471
##  3rd Qu.:0.005457   3rd Qu.:0.10321   3rd Qu.:0.06073   3rd Qu.:0.9061
##  Max.   :0.158219   Max.   :0.16016   Max.   :1.00000   Max.   :1.4439
##      count
##  Min.   :  30.00
##  1st Qu.:  38.25
##  Median :  52.00
##  Mean   : 121.49
##  3rd Qu.:  81.50
```

```
##  Max.   :2363.00
##
## mining info:
##    data ntransactions support confidence
##  trans1         14935   0.002       0.05
##                                                                  call
##  apriori(data = trans1, parameter = list(supp = 0.002, conf = 0.05))
```

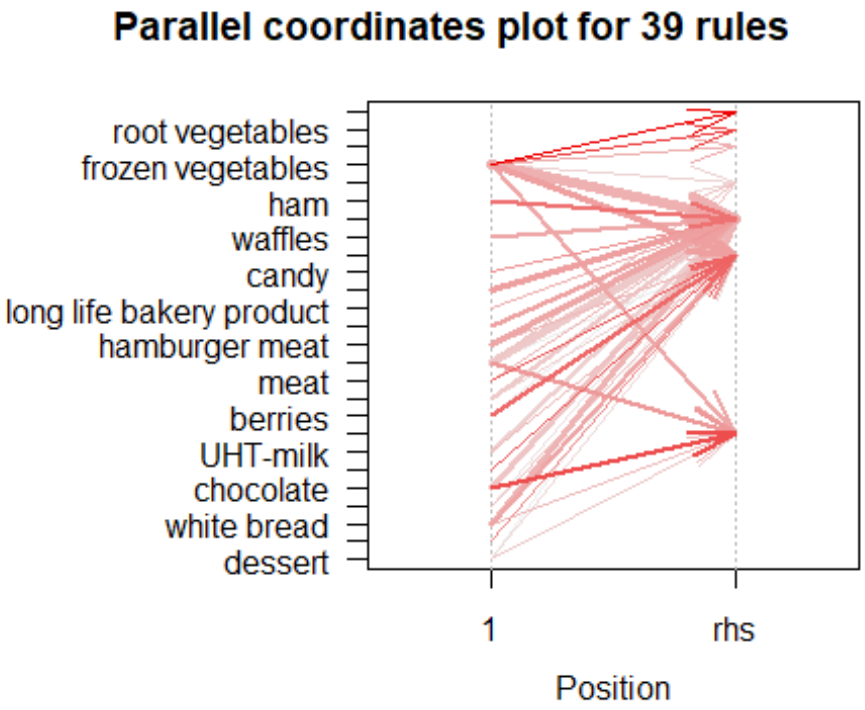
```
plot(rules.trans2, jitter = 0.25)
```




```
plot(rules.trans2[1:20], method="graph")
```

```
## Warning: ggrepel: 1 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```


```
plot(rules.trans2[1:50], method="paracoord")
```



When examining the co-occurrence of items in the dataset, we can see that there are many strong associations between items. For example, if a customer purchases citrus fruit, they are also likely to purchase tropical fruit, root vegetables, other vegetables, and whole milk. The lift values show us that these associations are stronger than what we would expect by chance.

The chi-squared test shows that there are significant associations between many of the items in the dataset. The p-values are very small, indicating that we can reject the null hypothesis of independence and conclude that there are associations between items.

Overall, the analysis shows that there are many strong associations between items in the groceries dataset, and that these associations are significant. This information can be useful for retailers who want to optimize product placement, promotions, and recommendations for customers based on their purchase history.

I set the minimum support to 0.001 and the minimum confidence to 0.08. This means that we are only interested in rules with a support of at least 0.001 (i.e., the rule must appear in at least 0.1% of all transactions) and a confidence of at least 0.08 (i.e., the rule must be correct at least 8% of the time). Then tried it with minimum support to 0.002 and the minimum confidence to 0.05.

Then I use the inspect() function to generate a list of association rules based on the specified itemsets, and sort the rules by lift. Finally, convert the list of rules to a data frame and show the top 10 rules.

Rules for closed itemsets

```
trans1.closed<-apriori(trans1, parameter=list(target="closed frequent
itemsets", support=0.01))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##          NA    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen                  target  ext
##      10 closed frequent itemsets TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 149
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[168 item(s), 14935 transaction(s)] done [0.00s].
## sorting and recoding items ... [64 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## filtering closed item sets ... done [0.00s].
```

```
## sorting transactions ... done [0.00s].
## writing ... [69 set(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

inspect(head(trans1.closed))

##     items              support    count
## [1] {red/blush wine}   0.01051222 157
## [2] {herbs}            0.01057918 158
## [3] {processed cheese} 0.01017744 152
## [4] {soft cheese}      0.01004352 150
## [5] {white wine}       0.01171744 175
## [6] {cat food}         0.01185136 177

is.closed(trans1.closed)

##              {red/blush wine}                      {herbs}
##                          TRUE                         TRUE
##            {processed cheese}                 {soft cheese}
##                          TRUE                         TRUE
##                 {white wine}                    {cat food}
##                          TRUE                         TRUE
##                {chewing gum}             {hygiene articles}
##                          TRUE                         TRUE
##              {specialty bar}                       {candy}
##                          TRUE                         TRUE
##              {sliced cheese}                   {ice cream}
##                          TRUE                         TRUE
##                      {grapes}                        {oil}
##                          TRUE                         TRUE
##             {misc. beverages}                 {hard cheese}
##                          TRUE                         TRUE
##         {specialty chocolate}                       {meat}
##                          TRUE                         TRUE
##                   {beverages}                       {ham}
##                          TRUE                         TRUE
##                {frozen meals}                 {butter milk}
##                          TRUE                         TRUE
##                       {sugar}    {long life bakery product}
##                          TRUE                         TRUE
##                {salty snack}                     {waffles}
##                          TRUE                         TRUE
##                      {onions}                   {UHT-milk}
##                          TRUE                         TRUE
##                     {berries}              {hamburger meat}
##                          TRUE                         TRUE
##                     {dessert}                     {napkins}
##                          TRUE                         TRUE
##                {cream cheese}                   {chocolate}
##                          TRUE                         TRUE
##                 {white bread}                     {chicken}
```

```
##                           TRUE                           TRUE
##           {frozen vegetables}                       {coffee}
##                           TRUE                           TRUE
##                    {margarine}                         {beef}
##                           TRUE                           TRUE
##       {fruit/vegetable juice}                         {curd}
##                           TRUE                           TRUE
##                         {butter}                         {pork}
##                           TRUE                           TRUE
##               {domestic eggs}                 {brown bread}
##                           TRUE                           TRUE
##                   {newspapers}                 {frankfurter}
##                           TRUE                           TRUE
##          {whipped/sour cream}                 {bottled beer}
##                           TRUE                           TRUE
##               {shopping bags}                 {canned beer}
##                           TRUE                           TRUE
##                    {pip fruit}                       {pastry}
##                           TRUE                           TRUE
##                 {citrus fruit}                 {bottled water}
##                           TRUE                           TRUE
##                       {sausage}               {root vegetables}
##                           TRUE                           TRUE
##              {tropical fruit}                       {yogurt}
##                           TRUE                           TRUE
##                          {soda}                   {rolls/buns}
##                           TRUE                           TRUE
##            {other vegetables}                   {whole milk}
##                           TRUE                           TRUE
##           {whole milk,yogurt}               {soda,whole milk}
##                           TRUE                           TRUE
## {other vegetables,rolls/buns}       {rolls/buns,whole milk}
##                           TRUE                           TRUE
## {other vegetables,whole milk}
##                           TRUE
```

```
freq.closed<-eclat(trans1, parameter=list(supp=0.001, maxlen=15,
target="closed frequent itemsets"))
```

```
## Eclat
##
## parameter specification:
##  tidLists support minlen maxlen                    target  ext
##     FALSE   0.001      1     15 closed frequent itemsets TRUE
##
## algorithmic control:
##  sparse sort verbose
##       7   -2    TRUE
##
## Absolute minimum support count: 14
```

```
##
## create itemset ...
## set transactions ...[168 item(s), 14935 transaction(s)] done [0.00s].
## sorting and recoding items ... [149 item(s)] done [0.00s].
## creating sparse bit matrix ... [149 row(s), 14935 column(s)] done [0.00s].
## writing  ... [753 set(s)] done [0.01s].
## Creating S4 object  ... done [0.00s].

inspect(head(freq.closed))

##     items                              support     count
## [1] {frozen fish, whole milk}          0.001071309 16
## [2] {rolls/buns, seasonal products}    0.001004352 15
## [3] {pot plants, whole milk}           0.001004352 15
## [4] {other vegetables, pot plants}     0.001004352 15
## [5] {pasta, whole milk}                0.001071309 16
## [6] {pickled vegetables, whole milk} 0.001004352 15

freq.max<-eclat(trans1, parameter=list(supp=0.001, maxlen=15,
target="maximally frequent itemsets"))

## Eclat
##
## parameter specification:
##  tidLists support minlen maxlen                       target  ext
##     FALSE   0.001      1     15 maximally frequent itemsets TRUE
##
## algorithmic control:
##  sparse sort verbose
##       7   -2    TRUE
##
## Absolute minimum support count: 14
##
## create itemset ...
## set transactions ...[168 item(s), 14935 transaction(s)] done [0.00s].
## sorting and recoding items ... [149 item(s)] done [0.00s].
## creating sparse bit matrix ... [149 row(s), 14935 column(s)] done [0.00s].
## writing  ... [667 set(s)] done [0.01s].
## Creating S4 object  ... done [0.00s].

inspect(head(freq.max))

##     items                              support     count
## [1] {frozen fish, whole milk}          0.001071309 16
## [2] {rolls/buns, seasonal products}    0.001004352 15
## [3] {pot plants, whole milk}           0.001004352 15
## [4] {other vegetables, pot plants}     0.001004352 15
## [5] {pasta, whole milk}                0.001071309 16
## [6] {pickled vegetables, whole milk} 0.001004352 15
```

```
####similarity & dissimilarity
trans.sel<-trans1[,itemFrequency(trans1)>0.05] # selected transations
d.jac.i<-dissimilarity(trans.sel, which="items") # Jaccard as default
round(d.jac.i,2)

##                  bottled water citrus fruit other vegetables pastry
rolls/buns
## citrus fruit              0.98
## other vegetables          0.97         0.97
## pastry                    0.97         0.98             0.98
## rolls/buns                0.97         0.97             0.95   0.98
## root vegetables           0.97         0.98             0.97   0.98
0.97
## sausage                   0.97         0.99             0.97   0.97
0.97
## soda                      0.97         0.97             0.95   0.97
0.96
## tropical fruit            0.97         0.98             0.97   0.98
0.96
## whole milk                0.97         0.96             0.94   0.97
0.94
## yogurt                    0.97         0.96             0.96   0.97
0.96
##                  root vegetables sausage soda tropical fruit whole milk
## citrus fruit
## other vegetables
## pastry
## rolls/buns
## root vegetables
## sausage                     0.97
## soda                        0.97    0.96
## tropical fruit              0.97    0.98 0.97
## whole milk                  0.97    0.96 0.95           0.96
## yogurt                      0.97    0.96 0.97           0.96       0.95
```

REFERENCES

Data: https://www.kaggle.com/datasets/heeraldedhia/groceries-dataset