The Bread Basket - Association Rule Mining

Divij Pherwani

Contents

Association Rule Mining is an unsupervised machine learning algorithm. The technique utilizes the apriori algorithm. The goal is to discover the association between the objects in datasets and common trends in the transactions.

The dataset used for analysis is "The Bread Basket" from Kaggle. The dataset belongs to a bakery located in Edinburgh. The dataset has 20507 entries, over 9000 transactions, and 4 columns. The dataset has transactions of customers who ordered different items from this bakery online and the time period of the data is from 26-01-11 to 27-12-03. Link to Kaggle dataset: (https://www.kaggle.com/mittalvasu95/the-bread-basket). There was no prior analysis done in R earlier and dataset meets the requirement of the project.

Loading Libraries and Data

```r
# Loading all the libraries that will be used
library("plyr")
library(arules)
library(arulesViz)
library(tidyverse)
library(lubridate)
library(dplyr)
library(plyr)
library(readxl)
library(xlsx)
library(lubridate)
library(data.table)
library(splitstackshape)
setwd(dirname(rstudioapi::getSourceEditorContext()$path))
```

The Transaction column contains the transaction ID of each transaction. The Item column contains the item name. The date_time column contains the timestamp of transaction. The period_day contains information about the period of day i.e., morning, afternoon, evening and night. The weekday_weekend column contains information if the day of the week is a weekday or weekend.

```r
#Loading the dataset and displaying the header
df <- read.csv("bread basket.csv")
head(df)
```

```
##   Transaction          Item        date_time period_day weekday_weekend
## 1           1         Bread 30-10-2016 09:58    morning         weekend
## 2           2  Scandinavian 30-10-2016 10:05    morning         weekend
## 3           2  Scandinavian 30-10-2016 10:05    morning         weekend
## 4           3 Hot chocolate 30-10-2016 10:07    morning         weekend
## 5           3           Jam 30-10-2016 10:07    morning         weekend
## 6           3       Cookies 30-10-2016 10:07    morning         weekend
```

Exploratory Data Analysis

After loading the dataset, we can see that column Transaction is type numeric, column Item is type Character, column data_time is type Character, column period_day is type Character and column weekday_weekend is type Character.

```r
# Summary of the bakery basket data frame
summary(df)
```

```
##   Transaction       Item            date_time          period_day
##  Min.   :   1   Length:20507       Length:20507       Length:20507
##  1st Qu.:2552   Class :character   Class :character   Class :character
##  Median :5137   Mode  :character   Mode  :character   Mode  :character
##  Mean   :4976
##  3rd Qu.:7357
##  Max.   :9684
##  weekday_weekend
##  Length:20507
##  Class :character
##  Mode  :character
##
##
##
```

The dataset has 20507 entries and 5 columns.

```r
# Another diagnostic for bakery basket data frame
str(df)
```

```
## 'data.frame':    20507 obs. of  5 variables:
##  $ Transaction    : int  1 2 2 3 3 3 4 5 5 5 ...
##  $ Item           : chr  "Bread" "Scandinavian" "Scandinavian" "Hot chocolate" ...
##  $ date_time      : chr  "30-10-2016 09:58" "30-10-2016 10:05" "30-10-2016 10:05" "30-10-2016 10:07"
##  $ period_day     : chr  "morning" "morning" "morning" "morning" ...
##  $ weekday_weekend: chr  "weekend" "weekend" "weekend" "weekend" ...
```

The bakery sells 94 unique items to its customers.

```r
# List of Unique Items sold by the bakery
unique(df$Item)
```

```
##  [1] "Bread"                         "Scandinavian"
##  [3] "Hot chocolate"                 "Jam"
##  [5] "Cookies"                       "Muffin"
##  [7] "Coffee"                        "Pastry"
##  [9] "Medialuna"                     "Tea"
## [11] "Tartine"                       "Basket"
## [13] "Mineral water"                 "Farm House"
## [15] "Fudge"                         "Juice"
## [17] "Ella's Kitchen Pouches"        "Victorian Sponge"
## [19] "Frittata"                      "Hearty & Seasonal"
## [21] "Soup"                          "Pick and Mix Bowls"
## [23] "Smoothies"                     "Cake"
## [25] "Mighty Protein"                "Chicken sand"
## [27] "Coke"                          "My-5 Fruit Shoot"
## [29] "Focaccia"                      "Sandwich"
## [31] "Alfajores"                     "Eggs"
## [33] "Brownie"                       "Dulce de Leche"
## [35] "Honey"                         "The BART"
## [37] "Granola"                       "Fairy Doors"
## [39] "Empanadas"                     "Keeping It Local"
## [41] "Art Tray"                      "Bowl Nic Pitt"
## [43] "Bread Pudding"                 "Adjustment"
## [45] "Truffles"                      "Chimichurri Oil"
## [47] "Bacon"                         "Spread"
## [49] "Kids biscuit"                  "Siblings"
## [51] "Caramel bites"                 "Jammie Dodgers"
## [53] "Tiffin"                        "Olum & polenta"
## [55] "Polenta"                       "The Nomad"
## [57] "Hack the stack"                "Bakewell"
## [59] "Lemon and coconut"             "Toast"
## [61] "Scone"                         "Crepes"
## [63] "Vegan mincepie"                "Bare Popcorn"
## [65] "Muesli"                        "Crisps"
## [67] "Pintxos"                       "Gingerbread syrup"
## [69] "Panatone"                      "Brioche and salami"
## [71] "Afternoon with the baker"      "Salad"
## [73] "Chicken Stew"                  "Spanish Brunch"
## [75] "Raspberry shortbread sandwich" "Extra Salami or Feta"
## [77] "Duck egg"                      "Baguette"
## [79] "Valentine's card"              "Tshirt"
## [81] "Vegan Feast"                   "Postcard"
## [83] "Nomad bag"                     "Chocolates"
## [85] "Coffee granules "              "Drinking chocolate spoons "
## [87] "Christmas common"              "Argentina Night"
## [89] "Half slice Monster "           "Gift voucher"
## [91] "Cherry me Dried fruit"         "Mortimer"
## [93] "Raw bars"                      "Tacos/Fajita"
```

This analysis is obvious as the day of the week is either a weekday or weekend.

```r
# List of unique items in weekday_weekend colulmn
unique(df$weekday_weekend)
```

```
## [1] "weekend" "weekday"
```

This analysis is obvious as well, the periods of the day include morning, afternoon, evening and night.

```r
# List of unique items in period_day column
unique(df$period_day)
```

```
## [1] "morning"   "afternoon" "evening"   "night"
```

There are no NA values in the dataset. While checking we iterating through each row and each column to find NA.

```r
# Checking for Null/NA values in the dataset and there is none
df[rowSums(is.na(df)) > 0,]
```

```
## [1] Transaction     Item            date_time       period_day
## [5] weekday_weekend
## <0 rows> (or 0-length row.names)
```

```r
df[,colSums(is.na(df)) > 0]
```

```
## data frame with 0 columns and 20507 rows
```

Coffee is by far the most popular item sold by the bakery. The other most popular items include Bread, Tea and Cake. Coffee culture in Edinburgh could be explored further via https://www.scotsman.com/lifestyle/food-and-drink/exploring-edinburghs-coffee-culture-1480578.

```r
# Ten most popular items sold by the bakery
x <- as.data.frame(plyr::count(df, 'Item'))
x <- x %>% arrange(desc(freq))
x[1:10,]
```

```
##             Item freq
## 1         Coffee 5471
## 2          Bread 3325
## 3            Tea 1435
## 4           Cake 1025
## 5         Pastry  856
## 6       Sandwich  771
## 7      Medialuna  616
## 8  Hot chocolate  590
## 9        Cookies  540
## 10       Brownie  379
```

Morning and afternoon is the most popular time for the transaction in bakery. The evening is not that popular and the transaction during the night at the bakery are almost non existent.

```r
# Most popular period of day for bakery sale
y <- as.data.frame(plyr::count(df, 'period_day'))
y <- y %>% arrange(desc(freq))
y
```

```
##   period_day  freq
## 1  afternoon 11569
## 2    morning  8404
## 3    evening   520
## 4      night    14
```

The weekday has a higher frequency of transactions at the bakery than the weekend. It is not an apple to apple comparison. The frequency difference between weekend and weekday is not that significant.

```r
# Frequency of weekday or weekend transaction
z <- as.data.frame(plyr::count(df, 'weekday_weekend'))
z <- z %>% arrange(desc(freq))
z
```

```
##   weekday_weekend  freq
## 1         weekday 12807
## 2         weekend  7700
```

The date_time column in broken into further columns to analyze the day more in depth. The column is split into date column, year column, month column and day column. Below we can see the head of the data frame after creating new columns.

```r
# Breaking down date_time column in date column, time coulumn, year column, month column and day column
temp <- as.POSIXlt(df$date_time, format="%d-%m-%Y %H:%M")
df$year <- year(temp)
df$month <- month(temp)
df$date <- date(temp)
df$time <- as.ITime(temp, format = "%H:%M")
df$day <- weekdays(date(temp))
df$month <- month.abb[df$month]
head(df)
```

```
##   Transaction           Item        date_time period_day weekday_weekend year
## 1           1          Bread 30-10-2016 09:58    morning         weekend 2016
## 2           2   Scandinavian 30-10-2016 10:05    morning         weekend 2016
## 3           2   Scandinavian 30-10-2016 10:05    morning         weekend 2016
## 4           3 Hot chocolate 30-10-2016 10:07    morning         weekend 2016
## 5           3            Jam 30-10-2016 10:07    morning         weekend 2016
## 6           3        Cookies 30-10-2016 10:07    morning         weekend 2016
##   month       date     time    day
## 1   Oct 2016-10-30 09:58:00 Sunday
## 2   Oct 2016-10-30 10:05:00 Sunday
## 3   Oct 2016-10-30 10:05:00 Sunday
## 4   Oct 2016-10-30 10:07:00 Sunday
## 5   Oct 2016-10-30 10:07:00 Sunday
## 6   Oct 2016-10-30 10:07:00 Sunday
```

The dataset contains bakery transactions from 30-10-2016 till 09-04-2017. There are almost 160 dates of transaction provided in the dataset.

```r
# Dates analysis
unique(df$date)
```

```
##   [1] "2016-10-30" "2016-10-31" "2016-11-01" "2016-11-02" "2016-11-03"
##   [6] "2016-11-04" "2016-11-05" "2016-11-06" "2016-11-07" "2016-11-08"
##  [11] "2016-11-09" "2016-11-10" "2016-11-11" "2016-11-12" "2016-11-13"
##  [16] "2016-11-14" "2016-11-15" "2016-11-16" "2016-11-17" "2016-11-18"
##  [21] "2016-11-19" "2016-11-20" "2016-11-21" "2016-11-22" "2016-11-23"
##  [26] "2016-11-24" "2016-11-25" "2016-11-26" "2016-11-27" "2016-11-28"
##  [31] "2016-11-29" "2016-11-30" "2016-12-01" "2016-12-02" "2016-12-03"
##  [36] "2016-12-04" "2016-12-05" "2016-12-06" "2016-12-07" "2016-12-08"
##  [41] "2016-12-09" "2016-12-10" "2016-12-11" "2016-12-12" "2016-12-13"
##  [46] "2016-12-14" "2016-12-15" "2016-12-16" "2016-12-17" "2016-12-18"
##  [51] "2016-12-19" "2016-12-20" "2016-12-21" "2016-12-22" "2016-12-23"
##  [56] "2016-12-24" "2016-12-27" "2016-12-28" "2016-12-29" "2016-12-30"
##  [61] "2016-12-31" "2017-01-01" "2017-01-03" "2017-01-04" "2017-01-05"
##  [66] "2017-01-06" "2017-01-07" "2017-01-08" "2017-01-09" "2017-01-10"
##  [71] "2017-01-11" "2017-01-12" "2017-01-13" "2017-01-14" "2017-01-15"
##  [76] "2017-01-16" "2017-01-17" "2017-01-18" "2017-01-19" "2017-01-20"
##  [81] "2017-01-21" "2017-01-22" "2017-01-23" "2017-01-24" "2017-01-25"
##  [86] "2017-01-26" "2017-01-27" "2017-01-28" "2017-01-29" "2017-01-30"
##  [91] "2017-01-31" "2017-02-01" "2017-02-02" "2017-02-03" "2017-02-04"
##  [96] "2017-02-05" "2017-02-06" "2017-02-07" "2017-02-08" "2017-02-09"
## [101] "2017-02-10" "2017-02-11" "2017-02-12" "2017-02-13" "2017-02-14"
## [106] "2017-02-15" "2017-02-16" "2017-02-17" "2017-02-18" "2017-02-19"
## [111] "2017-02-20" "2017-02-21" "2017-02-22" "2017-02-23" "2017-02-24"
## [116] "2017-02-25" "2017-02-26" "2017-02-27" "2017-02-28" "2017-03-01"
## [121] "2017-03-02" "2017-03-03" "2017-03-04" "2017-03-05" "2017-03-06"
## [126] "2017-03-07" "2017-03-08" "2017-03-09" "2017-03-10" "2017-03-11"
## [131] "2017-03-12" "2017-03-13" "2017-03-14" "2017-03-15" "2017-03-16"
## [136] "2017-03-17" "2017-03-18" "2017-03-19" "2017-03-20" "2017-03-21"
## [141] "2017-03-22" "2017-03-23" "2017-03-24" "2017-03-25" "2017-03-26"
## [146] "2017-03-27" "2017-03-28" "2017-03-29" "2017-03-30" "2017-03-31"
## [151] "2017-04-01" "2017-04-02" "2017-04-03" "2017-04-04" "2017-04-05"
## [156] "2017-04-06" "2017-04-07" "2017-04-08" "2017-04-09"
```

```r
min(unique(df$date))
```

```
## [1] "2016-10-30"
```

```r
max(unique(df$date))
```

```
## [1] "2017-04-09"
```

The frequency of November is the highest, which would be month where bakery sold most items or there were most amount of transactions. October is the month with least transactions but it is understandable as the dataset begins from 30-10-2016. Most of the month was not documented.

```r
# Frequency of transaction per month for bakery
x <- as.data.frame(plyr::count(df, 'month'))
x <- x %>% arrange(desc(freq))
x
```

```
##   month freq
## 1   Nov 4436
## 2   Mar 3944
## 3   Feb 3906
## 4   Jan 3356
## 5   Dec 3339
## 6   Apr 1157
## 7   Oct  369
```

2017 is the year with highest frequency of transactions but it is an unfair comparison as the months documented in 2016 and 2017 are not equal. On the brighter side, bakery has stable transactions every month.

```r
# Frequency of transaction per year for bakery
y <- as.data.frame(plyr::count(df, 'year'))
y <- y %>% arrange(desc(freq))
y
```

```
##   year  freq
## 1 2017 12363
## 2 2016  8144
```

Weekdays have the highest frequency, there are five days in weekdays and 2 in weekend so it is understandable as well. Saturday is the most popular day of the week with bakery transactions.

```r
# Frequency of transaction per day of the week for bakery
z <- as.data.frame(plyr::count(df, 'day'))
z <- z %>% arrange(desc(freq))
z
```

```
##         day freq
## 1  Saturday 4605
## 2    Friday 3124
## 3    Sunday 3095
## 4  Thursday 2646
## 5   Tuesday 2392
## 6    Monday 2324
## 7 Wednesday 2321
```

Data Engineering

After the analysis from the dataset, I am removing the unnecessary columns from the dataset and keeping only the items. The items are not in a single row, the same transaction id is present in multiple rows with item. So there is a need to document all the items for a transaction id in a single row to perform an easier analysis.

```r
# create an empty data frame and feeding it with the filtered data from the earlier loaded data frame

colClasses = c("numeric", "character")
col.names = c("Transaction", "Items")
table <- read.table(text = "", colClasses = colClasses, col.names = col.names)


for (i in unique(df$Transaction))
{
  x <- df[df$Transaction == i,2]
  y <- ""
  for (z in x)
  {
    z <- trimws(z)
    z <- tolower(z)
    if (y == "")
    {
      y <- paste0(y, z)
    }
    else
    {
     y <- paste(y, z, sep = " , ")
    }
  }
  table[nrow(table)+1, ] <- list(i,y)
  }

head(table)
```

```
##   Transaction                          Items
## 1           1                          bread
## 2           2   scandinavian , scandinavian
## 3           3 hot chocolate , jam , cookies
## 4           4                         muffin
## 5           5       coffee , pastry , bread
## 6           6   medialuna , pastry , muffin
```

I remove the Transaction ID column and split the Items by "," into new columns.

```r
table <- cSplit(table, "Items", sep=",")
table <- table[,2:ncol(table)]
head(table)
```

```
##         Items_01     Items_02 Items_03 Items_04 Items_05 Items_06 Items_07
## 1:         bread         <NA>     <NA>     <NA>     <NA>     <NA>     <NA>
## 2:  scandinavian scandinavian     <NA>     <NA>     <NA>     <NA>     <NA>
## 3: hot chocolate          jam  cookies     <NA>     <NA>     <NA>     <NA>
## 4:        muffin         <NA>     <NA>     <NA>     <NA>     <NA>     <NA>
## 5:        coffee       pastry    bread     <NA>     <NA>     <NA>     <NA>
## 6:     medialuna       pastry   muffin     <NA>     <NA>     <NA>     <NA>
##    Items_08 Items_09 Items_10 Items_11
## 1:     <NA>     <NA>     <NA>     <NA>
## 2:     <NA>     <NA>     <NA>     <NA>
```

```
## 3:     <NA>     <NA>     <NA>     <NA>
## 4:     <NA>     <NA>     <NA>     <NA>
## 5:     <NA>     <NA>     <NA>     <NA>
## 6:     <NA>     <NA>     <NA>     <NA>
```

The engineered data is stored in a csv file which will be later analyzed for association rule mining in transaction.

```
# Writing to the new basket file
write.table(table, "basket.csv", col.names = FALSE, row.names=FALSE, na = "", sep = ",")
```

Association Rule Mining

I begin by the reading the file as a transaction data and creating the transactional object.

```
bakery <- read.transactions("basket.csv", sep = ",")
```

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

```
summary(bakery)
```

```
## transactions as itemMatrix in sparse format with
##  9465 rows (elements/itemsets/transactions) and
##  94 columns (items) and a density of 0.02122827
##
## most frequent items:
##  coffee   bread     tea    cake  pastry (Other)
##    4528    3097    1350     983     815    8114
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10
## 3948 3059 1471  662  234   64   17    4    5    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   2.000   1.995   3.000  10.000
##
## includes extended item information - examples:
##                     labels
## 1               adjustment
## 2 afternoon with the baker
## 3                alfajores
```

Below we can see the first ten elements of the sparse matrix.

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

```
##      items
## [1]  {bread}
## [2]  {scandinavian}
```

```
## [3]  {cookies,hot chocolate,jam}
## [4]  {muffin}
## [5]  {bread,coffee,pastry}
## [6]  {medialuna,muffin,pastry}
## [7]  {coffee,medialuna,pastry,tea}
## [8]  {bread,pastry}
## [9]  {bread,muffin}
## [10] {medialuna,scandinavian}
```

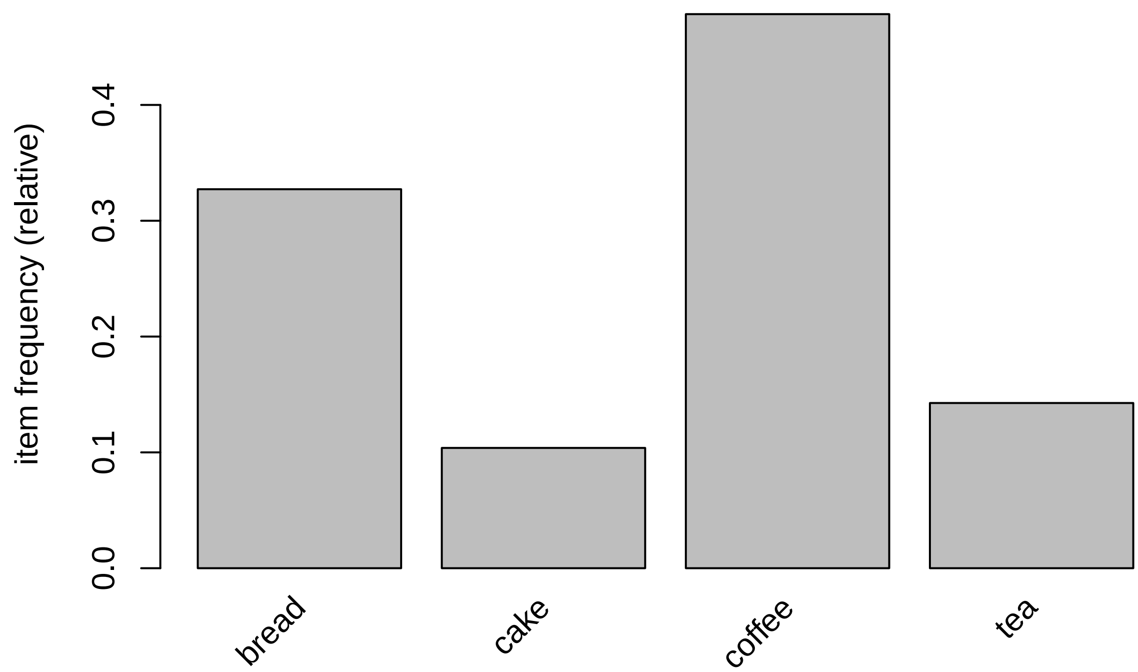
Checking the support level of first 5 items in bakery data

```
itemFrequency(bakery[,1:5])
```

```
##              adjustment afternoon with the baker                alfajores
##            0.0001056524             0.0045430534             0.0363444268
##         argentina night                 art tray
##            0.0007395668             0.0040147913
```

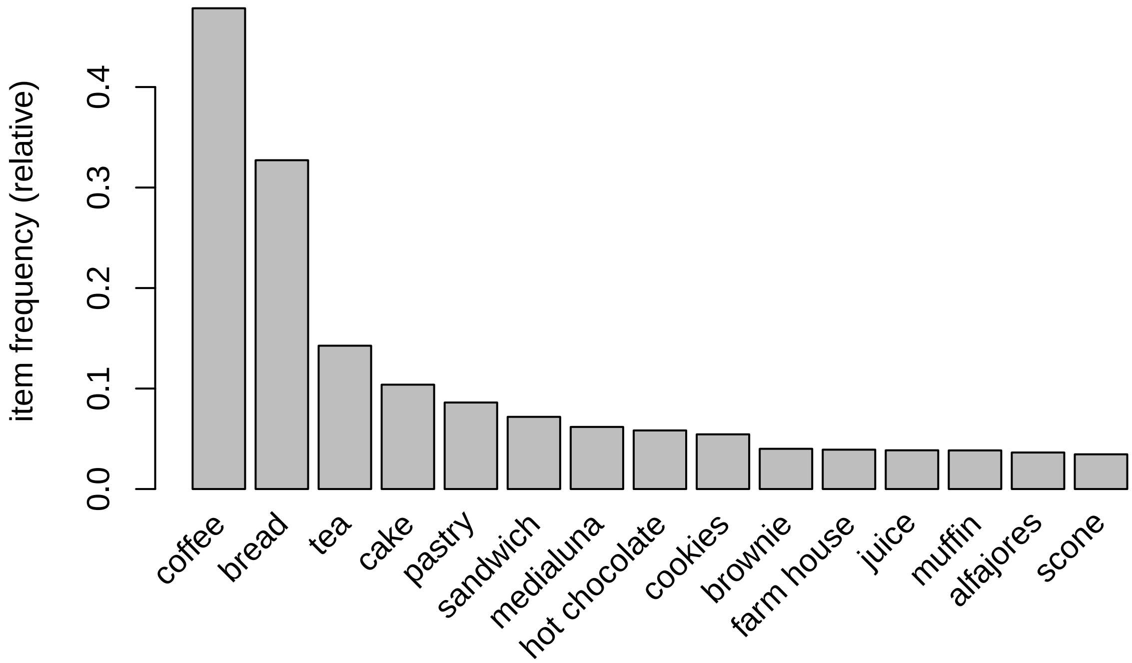
Frequency plot with set value of support at 10%

```
itemFrequencyPlot(bakery, support = 0.1)
```



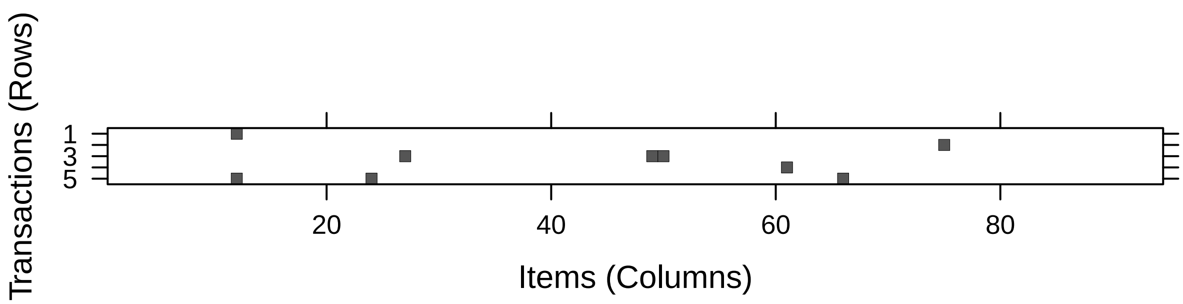
Plot of top 15 items

```
itemFrequencyPlot(bakery, topN = 15)
```



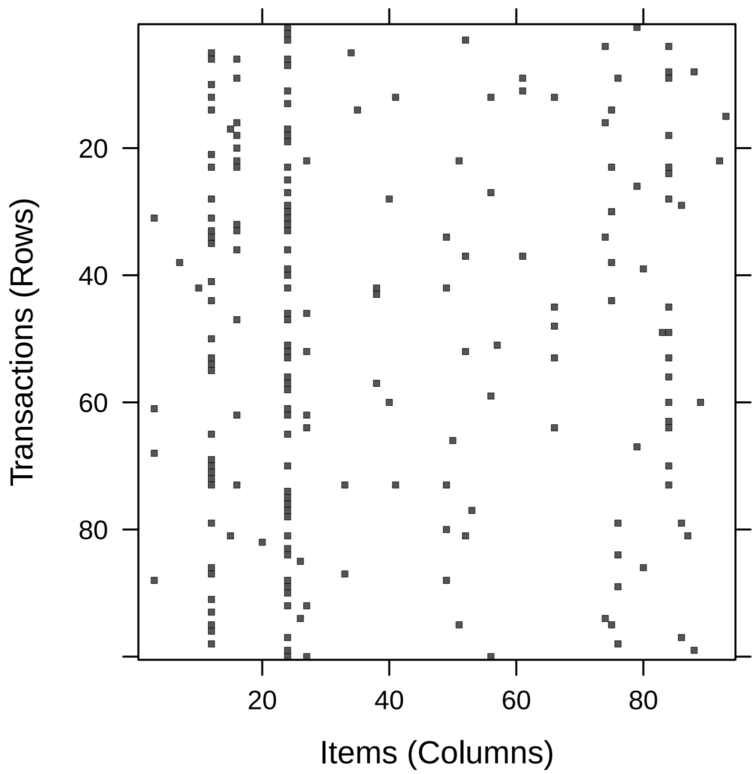
Below is the visualization of Sparse Matrix for first 5 transactions.

```
image(bakery[1:5])
```



Randomly selecting 100 transaction samples for visualization of sparse matrix

```
image(sample(bakery, 100))
```



Using the apriori algorithm I find the association rules, the support is set at 1% and confidence is set at 25% with minimum length of rule being 2.

```
bakeryrules <- apriori(bakery, parameter = list(support = 0.01, 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    0.01      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: 94
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[94 item(s), 9465 transaction(s)] done [0.00s].
## sorting and recoding items ... [30 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [24 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
bakeryrules
```

```
## set of 24 rules
```

Below is the summary of the association rules.

```r
summary(bakeryrules)
```

```
## set of 24 rules
##
## rule length distribution (lhs + rhs):sizes
##  2  3
## 21  3
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   2.000   2.125   2.000   3.000
##
## summary of quality measures:
##     support          confidence        coverage            lift
##  Min.   :0.01004   Min.   :0.2660   Min.   :0.01817   Min.   :0.5751
##  1st Qu.:0.01366   1st Qu.:0.3469   1st Qu.:0.03452   1st Qu.:0.8620
##  Median :0.01965   Median :0.4899   Median :0.04004   Median :1.0304
##  Mean   :0.02640   Mean   :0.4517   Mean   :0.06397   Mean   :1.0018
##  3rd Qu.:0.03098   3rd Qu.:0.5328   3rd Qu.:0.06432   3rd Qu.:1.1138
##  Max.   :0.09002   Max.   :0.7044   Max.   :0.32721   Max.   :1.4724
##      count
##  Min.   : 95.0
##  1st Qu.:129.2
##  Median :186.0
##  Mean   :249.8
##  3rd Qu.:293.2
##  Max.   :852.0
##
## mining info:
##    data ntransactions support confidence
##  bakery          9465    0.01       0.25
```

Inspecting first 10 bakery rules

```r
inspect(bakeryrules[1:10])
```

```
##      lhs                  rhs      support    confidence coverage   lift
## [1]  {spanish brunch} => {coffee} 0.01088220 0.5988372  0.01817221 1.2517655
## [2]  {toast}          => {coffee} 0.02366614 0.7044025  0.03359746 1.4724315
## [3]  {scone}          => {coffee} 0.01806656 0.5229358  0.03454834 1.0931067
## [4]  {soup}           => {coffee} 0.01584786 0.4601227  0.03444268 0.9618068
## [5]  {muffin}         => {coffee} 0.01880613 0.4890110  0.03845747 1.0221928
## [6]  {alfajores}      => {bread}  0.01035394 0.2848837  0.03634443 0.8706569
## [7]  {alfajores}      => {coffee} 0.01965135 0.5406977  0.03634443 1.1302349
## [8]  {brownie}        => {bread}  0.01077655 0.2691293  0.04004226 0.8225085
## [9]  {brownie}        => {coffee} 0.01965135 0.4907652  0.04004226 1.0258596
```

```
## [10] {juice}          => {coffee} 0.02060222 0.5342466  0.03856313 1.1167500
##      count
## [1]  103
## [2]  224
## [3]  171
## [4]  150
## [5]  178
## [6]   98
## [7]  186
## [8]  102
## [9]  186
## [10] 195
```

Inspecting first 5 bakery rules with decreasing order of lift

```r
inspect(sort(bakeryrules, by = "lift")[1:5])
```

```
##     lhs                 rhs      support    confidence coverage   lift
## [1] {toast}          => {coffee} 0.02366614 0.7044025  0.03359746 1.472431
## [2] {spanish brunch} => {coffee} 0.01088220 0.5988372  0.01817221 1.251766
## [3] {medialuna}      => {coffee} 0.03518225 0.5692308  0.06180666 1.189878
## [4] {pastry}         => {coffee} 0.04754358 0.5521472  0.08610671 1.154168
## [5] {alfajores}      => {coffee} 0.01965135 0.5406977  0.03634443 1.130235
##     count
## [1] 224
## [2] 103
## [3] 333
## [4] 450
## [5] 186
```

Inspecting first 5 bakery rules with decreasing order of confidence

```r
inspect(sort(bakeryrules, by = "confidence")[1:5])
```

```
##     lhs                 rhs      support    confidence coverage   lift
## [1] {toast}          => {coffee} 0.02366614 0.7044025  0.03359746 1.472431
## [2] {spanish brunch} => {coffee} 0.01088220 0.5988372  0.01817221 1.251766
## [3] {medialuna}      => {coffee} 0.03518225 0.5692308  0.06180666 1.189878
## [4] {pastry}         => {coffee} 0.04754358 0.5521472  0.08610671 1.154168
## [5] {alfajores}      => {coffee} 0.01965135 0.5406977  0.03634443 1.130235
##     count
## [1] 224
## [2] 103
## [3] 333
## [4] 450
## [5] 186
```

Inspecting first 5 bakery rules with decreasing order of support

```r
inspect(sort(bakeryrules, by = "support")[1:5])
```

```
##     lhs           rhs      support    confidence coverage   lift      count
## [1] {bread}    => {coffee} 0.09001585 0.2751049  0.32720549 0.5750592 852
## [2] {cake}     => {coffee} 0.05472795 0.5269583  0.10385631 1.1015151 518
## [3] {tea}      => {coffee} 0.04986793 0.3496296  0.14263074 0.7308402 472
## [4] {pastry}   => {coffee} 0.04754358 0.5521472  0.08610671 1.1541682 450
## [5] {sandwich} => {coffee} 0.03824617 0.5323529  0.07184363 1.1127916 362
```

Inspecting first 5 bakery rules with decreasing order of count

```
inspect(sort(bakeryrules, by = "count")[1:5])
```

```
##     lhs           rhs      support    confidence coverage   lift      count
## [1] {bread}    => {coffee} 0.09001585 0.2751049  0.32720549 0.5750592 852
## [2] {cake}     => {coffee} 0.05472795 0.5269583  0.10385631 1.1015151 518
## [3] {tea}      => {coffee} 0.04986793 0.3496296  0.14263074 0.7308402 472
## [4] {pastry}   => {coffee} 0.04754358 0.5521472  0.08610671 1.1541682 450
## [5] {sandwich} => {coffee} 0.03824617 0.5323529  0.07184363 1.1127916 362
```

Preparing and inspecting rules for Coffee by setting Consequent as Coffee.

```
rules.coffee<-apriori(data=bakery, parameter=list(supp=0.01,conf = 0.25), appearance=list(default="lhs"
rules.coffee.byconf<-sort(rules.coffee, by="confidence", decreasing=TRUE)
inspect(head(rules.coffee.byconf))
```

```
##     lhs                 rhs      support    confidence coverage   lift
## [1] {toast}          => {coffee} 0.02366614 0.7044025  0.03359746 1.472431
## [2] {spanish brunch} => {coffee} 0.01088220 0.5988372  0.01817221 1.251766
## [3] {medialuna}      => {coffee} 0.03518225 0.5692308  0.06180666 1.189878
## [4] {pastry}         => {coffee} 0.04754358 0.5521472  0.08610671 1.154168
## [5] {alfajores}      => {coffee} 0.01965135 0.5406977  0.03634443 1.130235
## [6] {juice}          => {coffee} 0.02060222 0.5342466  0.03856313 1.116750
##     count
## [1] 224
## [2] 103
## [3] 333
## [4] 450
## [5] 186
## [6] 195
```

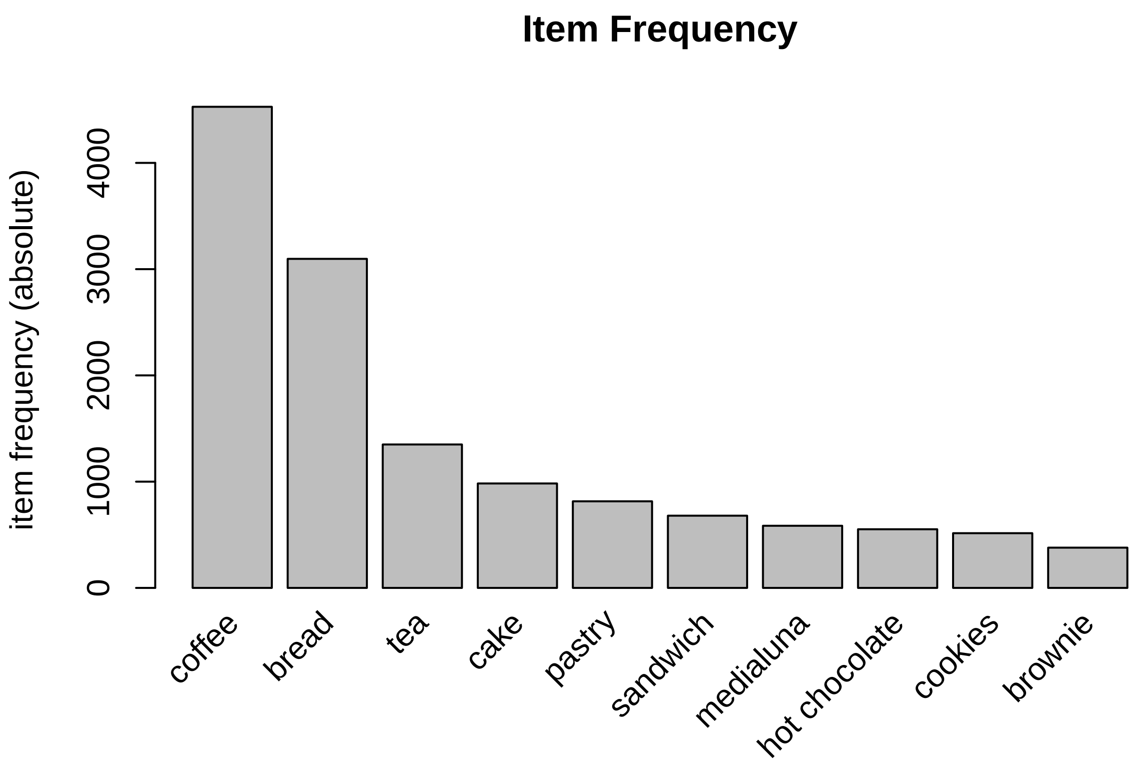
Using another way to filter rules and inspecting all rules for Bread

```
breadrules <- subset(bakeryrules, items %in% "bread")
inspect(breadrules)
```

```
##     lhs               rhs      support    confidence coverage   lift      count
## [1] {alfajores}    => {bread}  0.01035394 0.2848837  0.03634443 0.8706569  98
## [2] {brownie}      => {bread}  0.01077655 0.2691293  0.04004226 0.8225085 102
## [3] {cookies}      => {bread}  0.01447438 0.2660194  0.05441099 0.8130041 137
## [4] {medialuna}    => {bread}  0.01690438 0.2735043  0.06180666 0.8358792 160
## [5] {pastry}       => {bread}  0.02916006 0.3386503  0.08610671 1.0349774 276
## [6] {bread}        => {coffee} 0.09001585 0.2751049  0.32720549 0.5750592 852
## [7] {bread,pastry} => {coffee} 0.01119915 0.3840580  0.02916006 0.8028067 106
## [8] {bread,cake}   => {coffee} 0.01003698 0.4298643  0.02334918 0.8985568  95
```

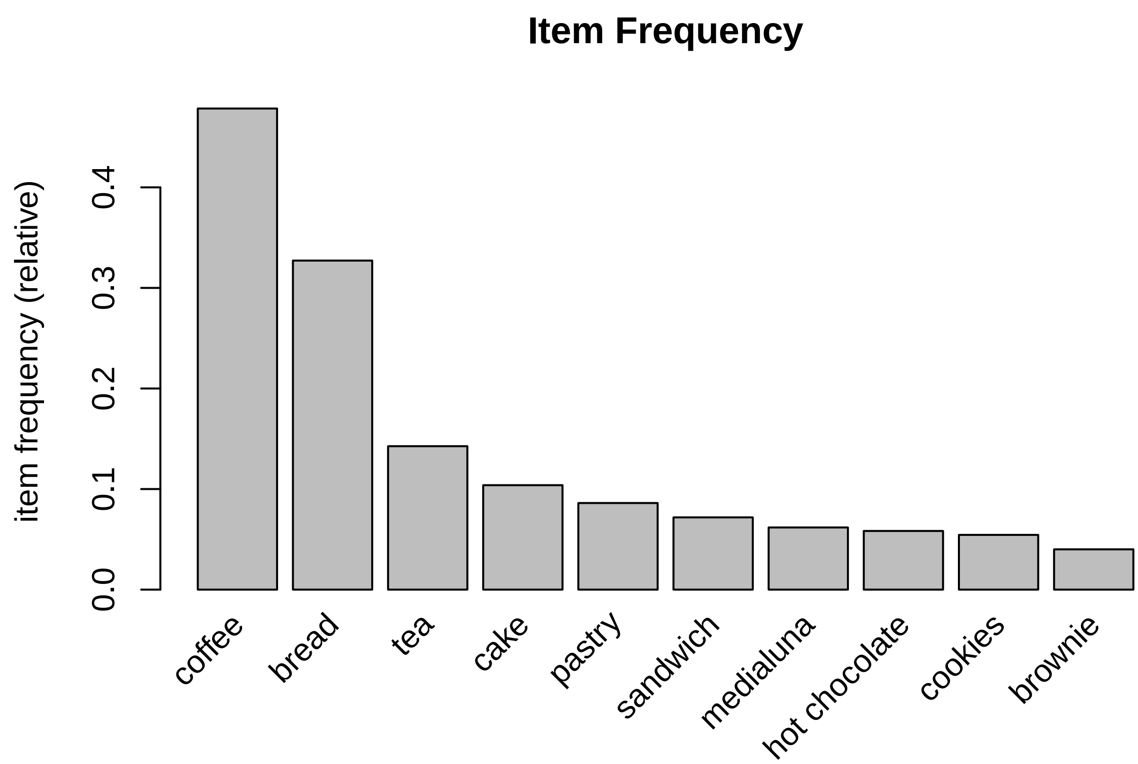
I observed earlier during the EDA that coffee is the most popular item. Using the itemFrequencyPlot function I plot the item frequency. The plot below is absolute item frequency.

```
itemFrequencyPlot(bakery, topN=10, type="absolute", main="Item Frequency")
```



I observed earlier during the EDA that coffee is the most popular item. Using the itemFrequencyPlot function I plot the item frequency. The plot below is relative item frequency.

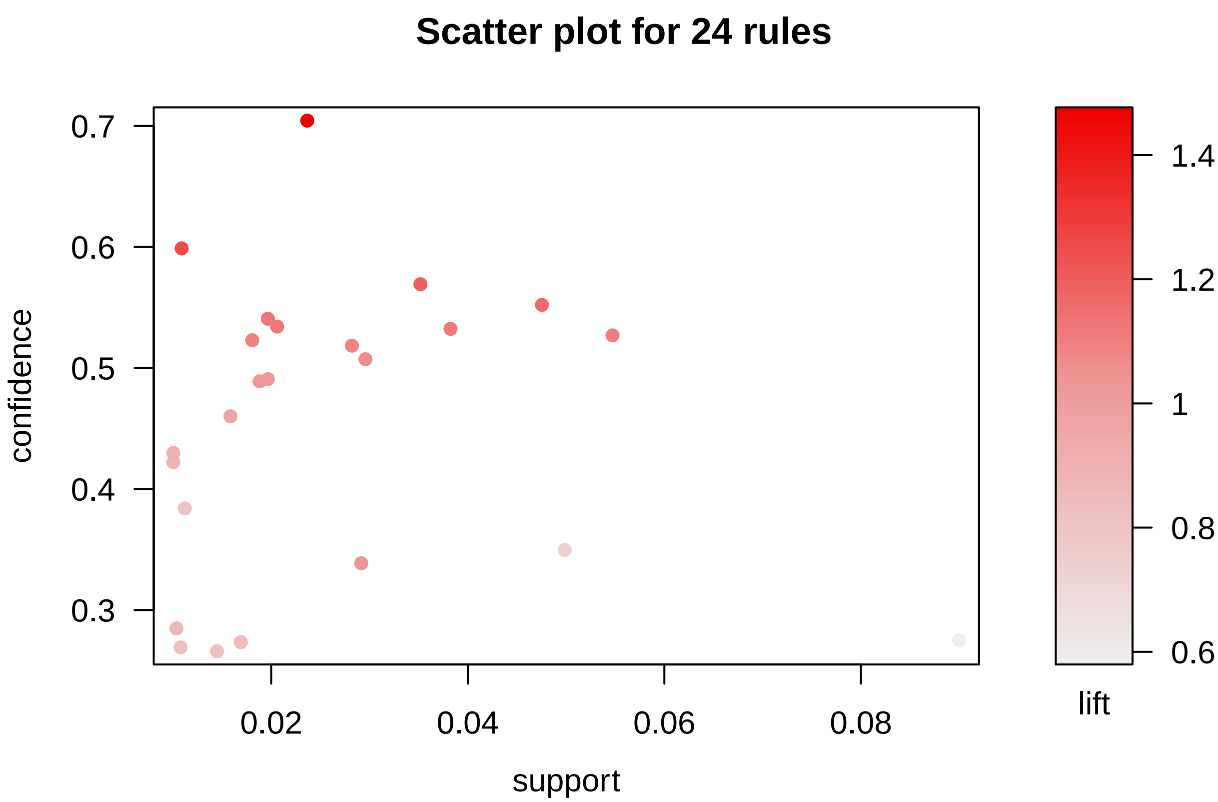
```
itemFrequencyPlot(bakery, topN=10, type="relative", main="Item Frequency")
```



Below is the scatter plot for all the 24 rules. On the x-axis, we have the support. On the y-axis, we have the confidence. The intensity of color in the plot signifies the lift value (darker the higher).

The relationship is not clear by the plot, there seems to be a correlation between confidence and lift.

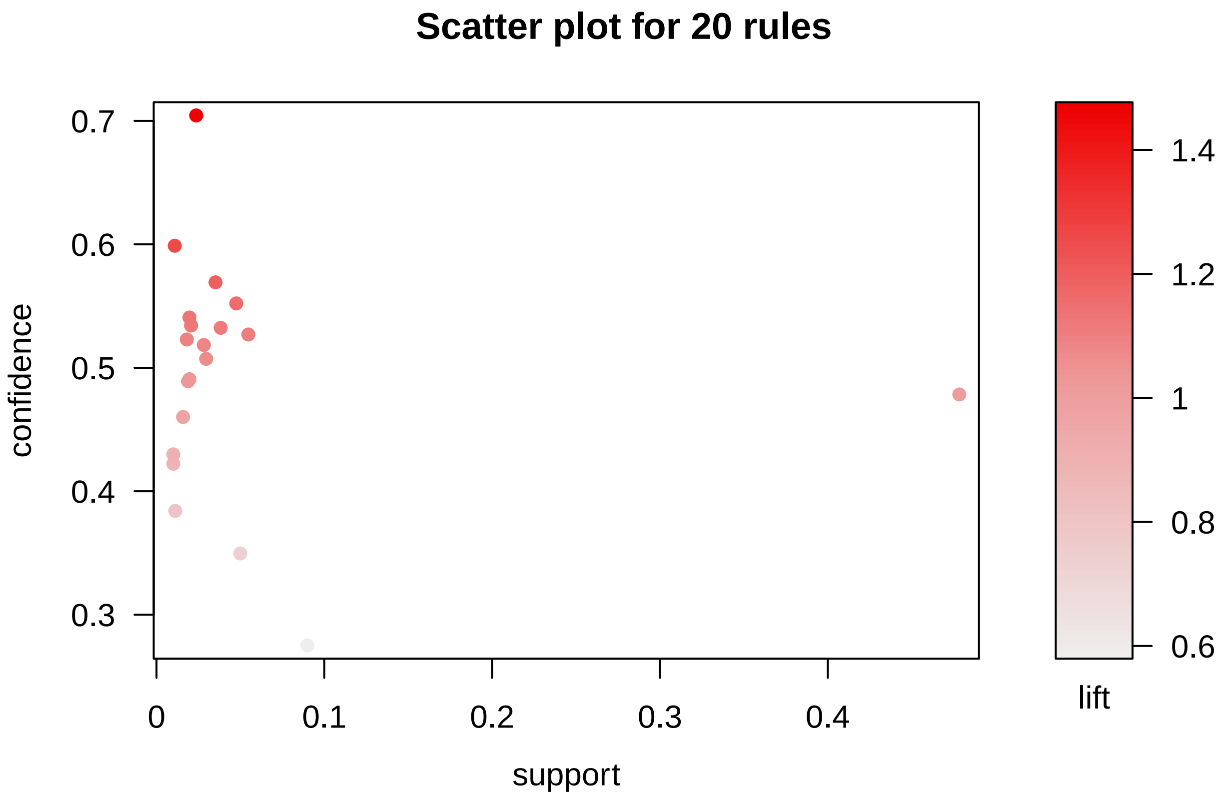
```
plot(bakeryrules)
```



Below is the scatter plot for all rules for item coffee. On the x-axis, we have the support. On the y-axis, we have the confidence. The intensity of color in the plot signifies the lift value (darker the higher).

Higher the confidence, higher the lift.

```
plot(rules.coffee)
```

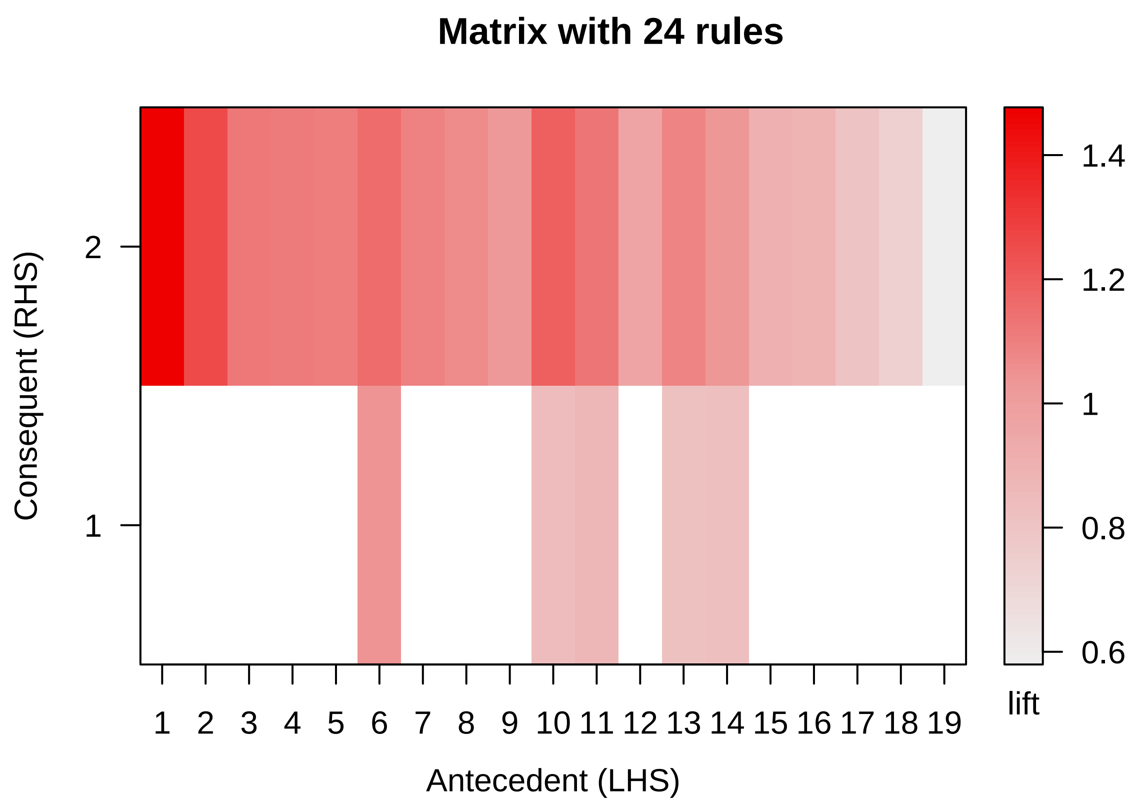


Below is the matrix plot for all bakery rules. On the x-axis, we have the Antecendent or LHS. On the y-axis, we have the Consequent or RHS. The intensity of color in the plot signifies the lift value (darker the higher).

Higher the Consequent, higher the lift. Higher the Antecedent, lower the lift.

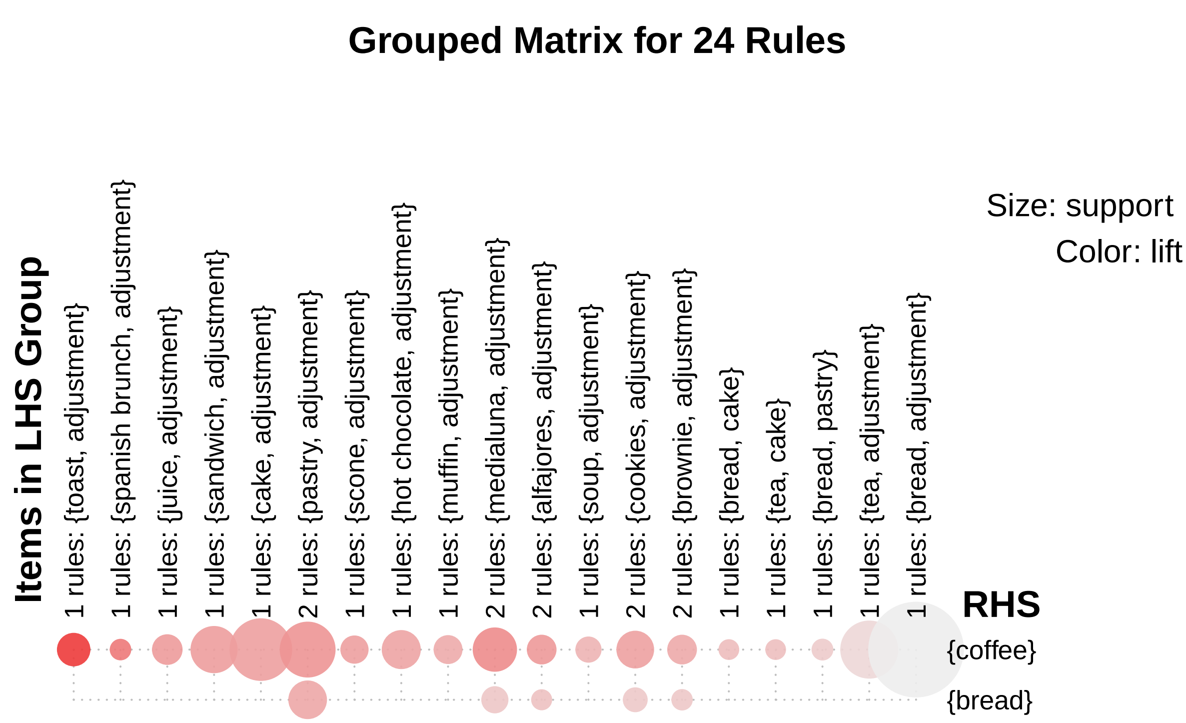
```
plot(bakeryrules, method="matrix", measure="lift")
```

```
## Itemsets in Antecedent (LHS)
##  [1] "{toast}"           "{spanish brunch}" "{juice}"          "{sandwich}"
##  [5] "{cake}"            "{pastry}"         "{scone}"          "{hot chocolate}"
##  [9] "{muffin}"          "{medialuna}"      "{alfajores}"      "{soup}"
## [13] "{cookies}"         "{brownie}"        "{bread,cake}"     "{cake,tea}"
## [17] "{bread,pastry}"    "{tea}"            "{bread}"
## Itemsets in Consequent (RHS)
## [1] "{bread}"  "{coffee}"
```



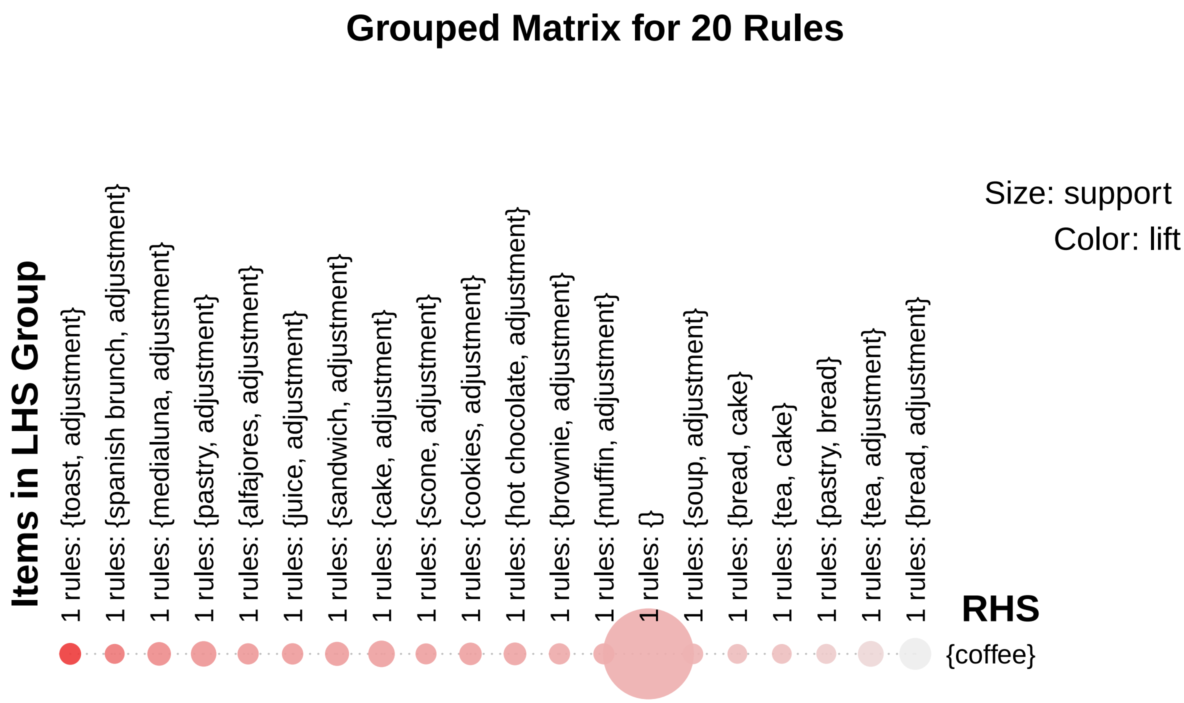
The grouped plot is presented below for all rules. On the RHS, we have Consequent and on the LHS we have Antecedent. The size of the circles represent the support and the intensity of the color represent the lift.

```
plot(bakeryrules, method="grouped")
```



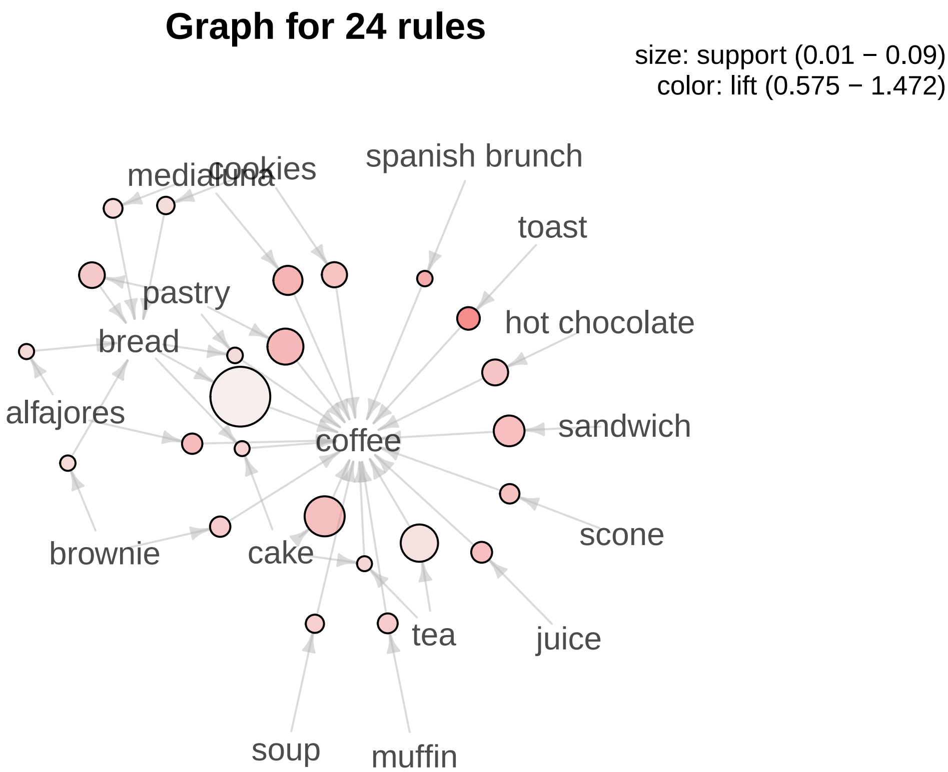
The grouped plot is presented below for all coffee rules which are above 80% of the total. On the RHS, we have Consequent and on the LHS we have Antecedent. The size of the circles represent the support and the intensity of the color represent the lift.

```
plot(rules.coffee, method="grouped")
```



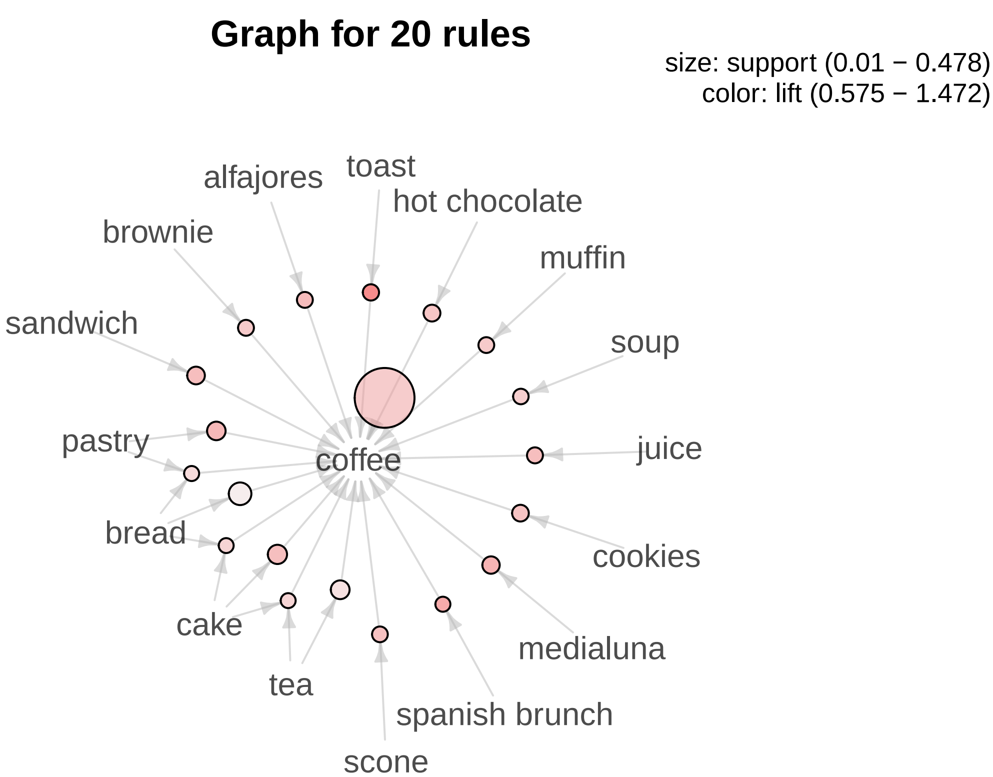
I plot all the 24 rules for bakery for vizualization

```
plot(bakeryrules, method="graph")
```



I plot all the 20 rules for coffee in bakery for vizualization

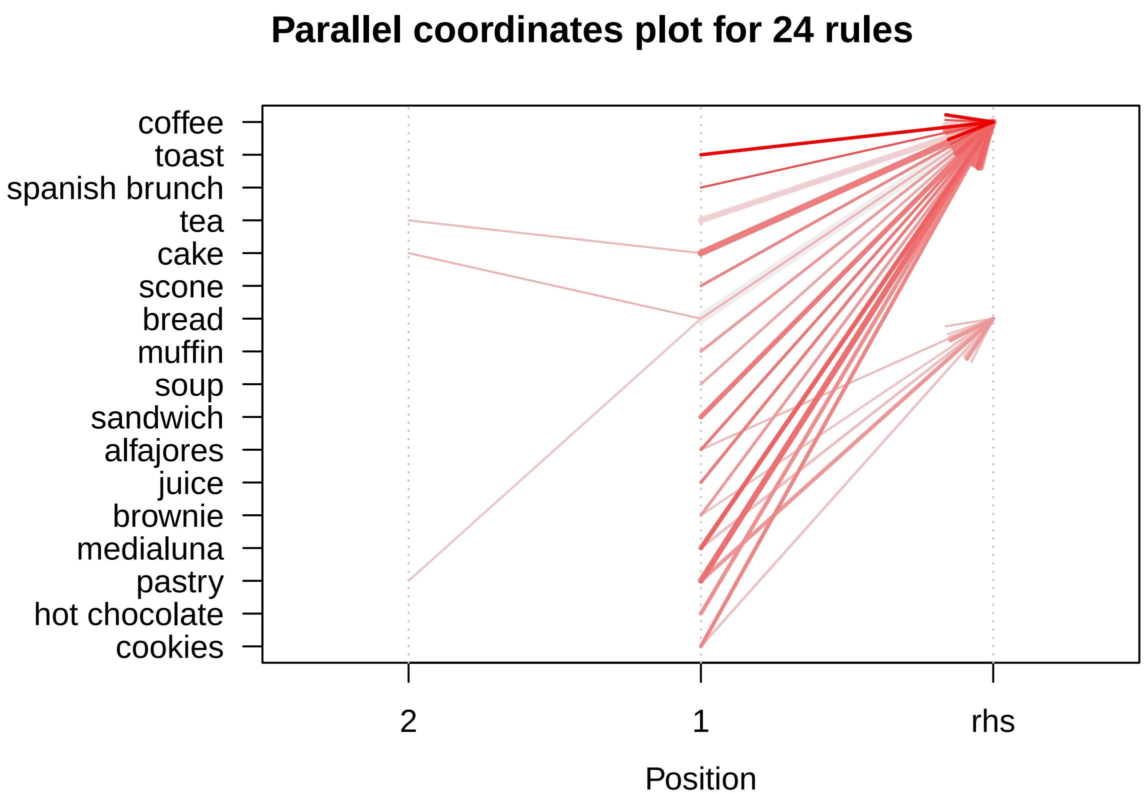
```r
plot(rules.coffee, method="graph")
```



Below we have Parallel coordinates plot for all the bakery rules. On the x-axis, we have position in the rule.

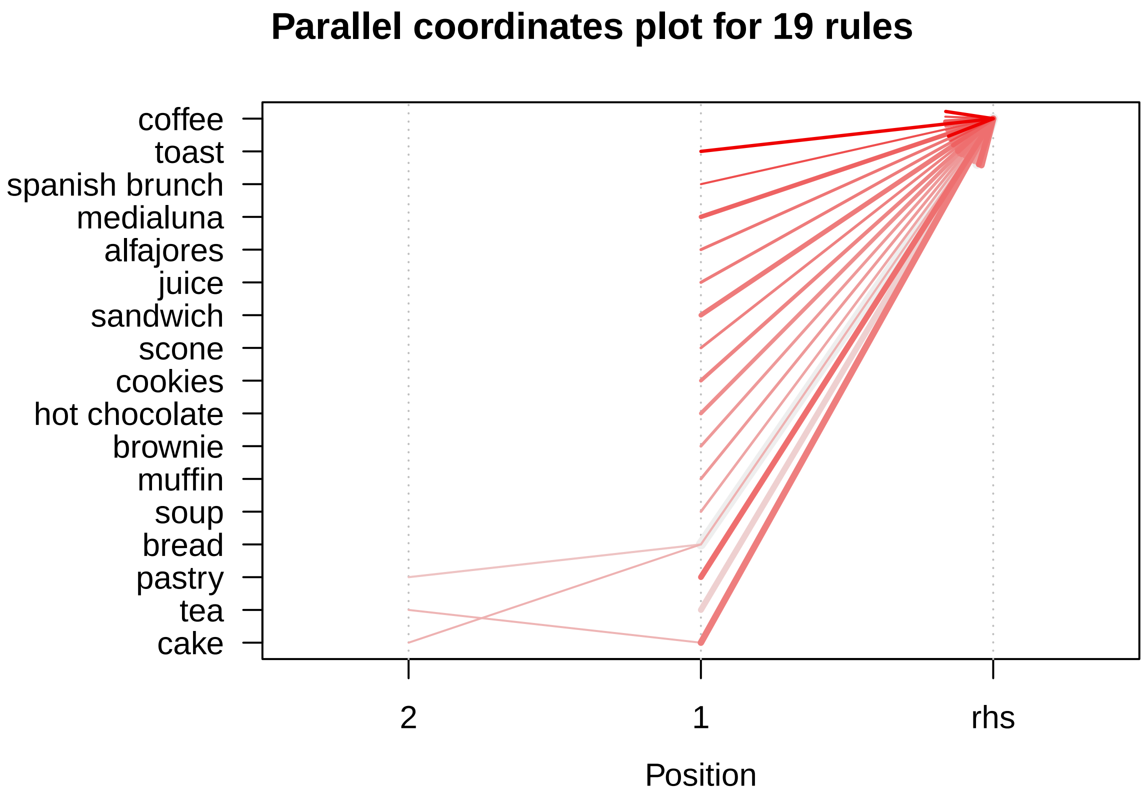
On the y-axis, we have the nominal values. The support is represented by the width of the arrow line and the confidence is represented by the intensity of the color.

```
plot(bakeryrules, method="paracoord", control=list(reorder=TRUE))
```



Below we have Parallel coordinates plot for item coffee. On the x-axis, we have position in the rule. On the y-axis, we have the nominal values. The support is represented by the width of the arrow line and the confidence is represented by the intensity of the color.

```
plot(rules.coffee, method="paracoord", control=list(reorder=TRUE))
```



Below is an interactive chart to project the association rules network for relationship between rules and items.

```
plot(bakeryrules, method = "graph", measure = "lift", shading = "confidence", engine = "htmlwidget")
```

Using eclat algorithm, we perform basic statistics with reference to confidence. The minimum support is set at 1 % with 5 items or less.

```
freq.items<-eclat(bakery, parameter=list(supp=0.01, maxlen=5))
```

```
## Eclat
##
## parameter specification:
##  tidLists support minlen maxlen            target  ext
##     FALSE    0.01      1      5 frequent itemsets TRUE
##
## algorithmic control:
##  sparse sort verbose
##       7   -2    TRUE
##
## Absolute minimum support count: 94
##
## create itemset ...
## set transactions ...[94 item(s), 9465 transaction(s)] done [0.00s].
## sorting and recoding items ... [30 item(s)] done [0.00s].
## creating sparse bit matrix ... [30 row(s), 9465 column(s)] done [0.00s].
## writing  ... [61 set(s)] done [0.00s].
## Creating S4 object  ... done [0.00s].
```

```r
inspect(freq.items)
```

```
##      items                   support    transIdenticalToItemsets count
## [1]  {coffee,spanish brunch} 0.01088220  103                       103
## [2]  {coffee,toast}          0.02366614  224                       224
## [3]  {coffee,scone}          0.01806656  171                       171
## [4]  {coffee,soup}           0.01584786  150                       150
## [5]  {coffee,muffin}         0.01880613  178                       178
## [6]  {alfajores,coffee}      0.01965135  186                       186
## [7]  {alfajores,bread}       0.01035394   98                        98
## [8]  {brownie,coffee}        0.01965135  186                       186
## [9]  {bread,brownie}         0.01077655  102                       102
## [10] {coffee,juice}          0.02060222  195                       195
## [11] {coffee,cookies}        0.02820919  267                       267
## [12] {bread,cookies}         0.01447438  137                       137
## [13] {coffee,medialuna}      0.03518225  333                       333
## [14] {bread,medialuna}       0.01690438  160                       160
## [15] {coffee,hot chocolate}  0.02958267  280                       280
## [16] {bread,hot chocolate}   0.01341786  127                       127
## [17] {cake,hot chocolate}    0.01141046  108                       108
## [18] {coffee,sandwich}       0.03824617  362                       362
## [19] {bread,sandwich}        0.01701004  161                       161
## [20] {sandwich,tea}          0.01436873  136                       136
## [21] {bread,coffee,pastry}   0.01119915  106                       106
## [22] {coffee,pastry}         0.04754358  450                       450
## [23] {bread,pastry}          0.02916006  276                       276
## [24] {cake,coffee,tea}       0.01003698   95                        95
## [25] {bread,cake,coffee}     0.01003698   95                        95
## [26] {cake,coffee}           0.05472795  518                       518
## [27] {bread,cake}            0.02334918  221                       221
## [28] {cake,tea}              0.02377179  225                       225
## [29] {coffee,tea}            0.04986793  472                       472
## [30] {bread,tea}             0.02810354  266                       266
## [31] {bread,coffee}          0.09001585  852                       852
## [32] {coffee}                0.47839408 4528                      4528
## [33] {bread}                 0.32720549 3097                      3097
## [34] {tea}                   0.14263074 1350                      1350
## [35] {cake}                  0.10385631  983                       983
## [36] {pastry}                0.08610671  815                       815
## [37] {sandwich}              0.07184363  680                       680
## [38] {hot chocolate}         0.05832013  552                       552
## [39] {medialuna}             0.06180666  585                       585
## [40] {cookies}               0.05441099  515                       515
## [41] {juice}                 0.03856313  365                       365
## [42] {brownie}               0.04004226  379                       379
## [43] {alfajores}             0.03634443  344                       344
## [44] {muffin}                0.03845747  364                       364
## [45] {soup}                  0.03444268  326                       326
## [46] {scone}                 0.03454834  327                       327
## [47] {toast}                 0.03359746  318                       318
## [48] {farm house}            0.03919704  371                       371
## [49] {truffles}              0.02028526  192                       192
## [50] {spanish brunch}        0.01817221  172                       172
```

```
## [51] {scandinavian}         0.02905441  275                         275
## [52] {coke}                 0.01944004  184                         184
## [53] {tiffin}               0.01542525  146                         146
## [54] {mineral water}        0.01415742  134                         134
## [55] {jammie dodgers}       0.01320655  125                         125
## [56] {chicken stew}         0.01299525  123                         123
## [57] {jam}                  0.01500264  142                         142
## [58] {salad}                0.01045959   99                          99
## [59] {fudge}                0.01500264  142                         142
## [60] {hearty & seasonal}    0.01056524  100                         100
## [61] {baguette}             0.01605917  152                         152
```

Using eclat algorithm, we perform basic statistics with reference to confidence. The minimum support is set at 5 % with 5 items or less.

```
freq.items<-eclat(bakery, parameter=list(supp=0.05, maxlen=5))
```

```
## Eclat
##
## parameter specification:
##  tidLists support minlen maxlen            target  ext
##     FALSE    0.05      1      5 frequent itemsets TRUE
##
## algorithmic control:
##  sparse sort verbose
##       7   -2    TRUE
##
## Absolute minimum support count: 473
##
## create itemset ...
## set transactions ...[94 item(s), 9465 transaction(s)] done [0.00s].
## sorting and recoding items ... [9 item(s)] done [0.00s].
## creating bit matrix ... [9 row(s), 9465 column(s)] done [0.00s].
## writing  ... [11 set(s)] done [0.00s].
## Creating S4 object  ... done [0.00s].
```

```
inspect(freq.items)
```

```
##      items           support    transIdenticalToItemsets count
## [1]  {cake,coffee}   0.05472795  518                      518
## [2]  {bread,coffee}  0.09001585  852                      852
## [3]  {coffee}        0.47839408 4528                     4528
## [4]  {bread}         0.32720549 3097                     3097
## [5]  {tea}           0.14263074 1350                     1350
## [6]  {cake}          0.10385631  983                      983
## [7]  {pastry}        0.08610671  815                      815
## [8]  {sandwich}      0.07184363  680                      680
## [9]  {hot chocolate} 0.05832013  552                      552
## [10] {medialuna}     0.06180666  585                      585
## [11] {cookies}       0.05441099  515                      515
```

Using the S4 method, we create association rules with minimum confidence set at 10 %.

```r
freq.rules<-ruleInduction(freq.items, bakery, confidence=0.1)
inspect(freq.rules)
```

```
##     lhs         rhs      support    confidence lift      itemset
## [1] {coffee} => {cake}   0.05472795 0.1143993  1.1015151 1
## [2] {cake}   => {coffee} 0.05472795 0.5269583  1.1015151 1
## [3] {coffee} => {bread}  0.09001585 0.1881625  0.5750592 2
## [4] {bread}  => {coffee} 0.09001585 0.2751049  0.5750592 2
```

The rules are saved in a csv file.

```r
# saving the output
write(bakeryrules, file = "bakeryrules.csv", sep = ",", quote = TRUE, row.names = FALSE)
```

Summary

I used the apriori algorithm for association mining to mine rules for the bakery. Coffee is by far the most popular item at the bakery. It was hard to analysze this data because a lot of people just buy a single item at the bakery. People who buy toast are likely buy coffee with 70% confidence by far the highest and 1.4 lift value. People who buy bread buy coffee with the highest count. Bakery is for bread products but definitely coffee is one item the shop can't get out of their menu. Also, plots were provided and along with complete analysis.

References:

1. Association Rule Mining in R, https://medium.com/swlh/association-rule-mining-in-r-acbd15e0de89
2. University of Warsaw, Unsupervised Learning Course by dr Jacek Lewkowicz