A Gentle Introduction on Market Basket Analysis - Association Rules

Introduction

Market Basket Analysis is one of the key techniques used by the large retailers that uncovers associations between items by looking for combinations of items that occur together frequently in transactions. In other words, it allows the retailers to identify relationships between the items that people buy.

Association Rules is widely used to analyze retail basket or transaction data, is intended to identify strong rules discovered in transaction data using some measures of interestingness, based on the concept of strong rules.

An Example of Association Rules

- Assume there are 100 customers
- 10 out of them bought milk, 8 bought butter and 6 bought both of them.
- bought milk => bought butter
- Support = P(Milk & Butter) = 6/100 = 0.06
- confidence = support/P(Butter) = 0.06/0.08 = 0.75
- lift = confidence/P(Milk) = 0.75/0.10 = 7.5

Note: this example is extremely small. In practice, a rule needs a support of several hundred transactions before it can be considered statistically significant, and datasets often contain thousands or millions of transactions.

Ok, enough for the theory, let's get to the code.

The dataset we are using today comes from UCI Machine Learning repository. The dataset is called Online Retail and can be found here. It contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered online retail.

Load the packages

Data preprocessing and exploring

```
## Observations: 406,829
## Variables: 10
## $ InvoiceNo   <dbl> 536365, 536365, 536365, 536365, 536365, 536365, 53...
## $ StockCode   <chr> "85123A", "71053", "84406B", "84029G", "84029E", "...
## $ Description <fct> WHITE HANGING HEART T-LIGHT HOLDER, WHITE METAL LA...
## $ Quantity    <dbl> 6, 6, 8, 6, 6, 2, 6, 6, 6, 32, 6, 6, 8, 6, 6, 3, 2...
## $ InvoiceDate <dttm> 2010-12-01 08:26:00, 2010-12-01 08:26:00, 2010-12...
## $ UnitPrice   <dbl> 2.55, 3.39, 2.75, 3.39, 3.39, 7.65, 4.25, 1.85, 1....
## $ CustomerID  <dbl> 17850, 17850, 17850, 17850, 17850, 17850, 17850, 1...
## $ Country     <fct> United Kingdom, United Kingdom, United Kingdom, Un...
## $ Date        <date> 2010-12-01, 2010-12-01, 2010-12-01, 2010-12-01, 2...
## $ Time        <chr> "08:26:00", "08:26:00", "08:26:00", "08:26:00", "0...
```

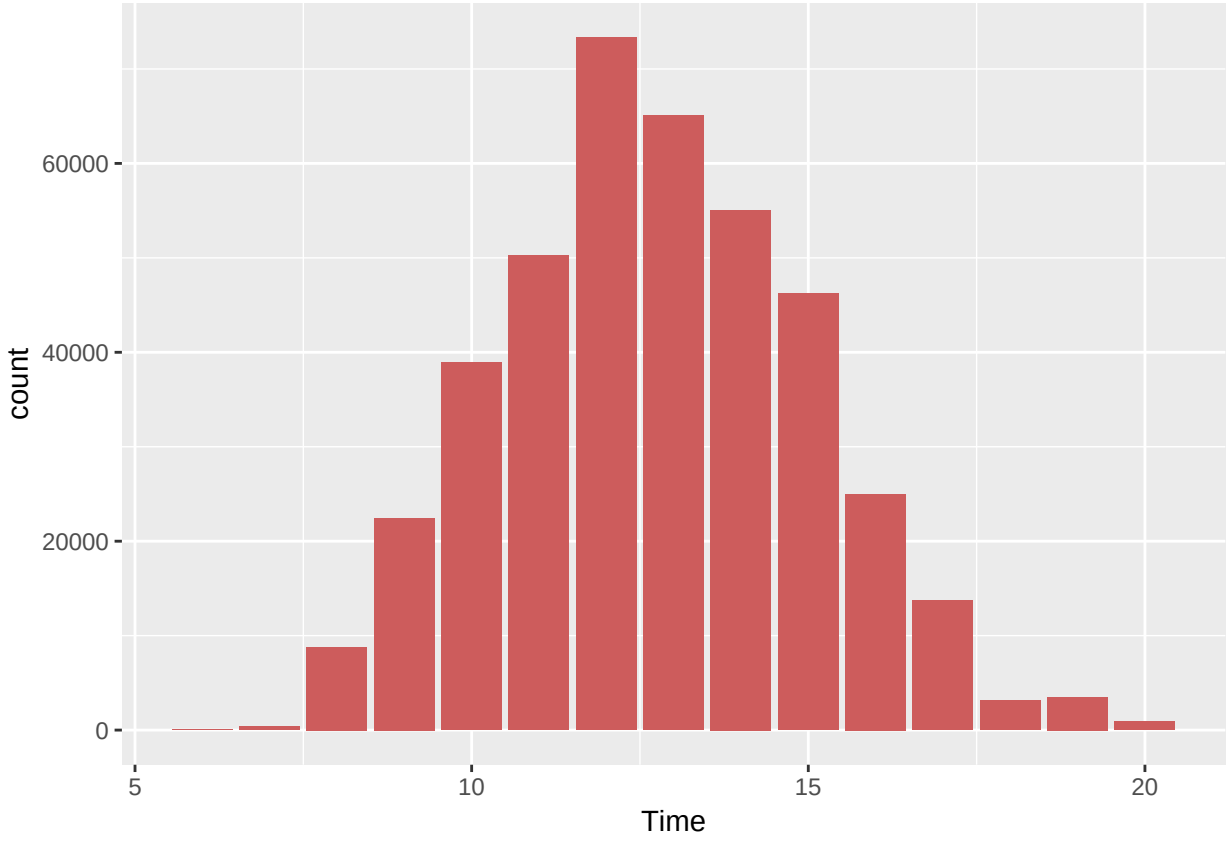
```
## Classes 'tbl_df', 'tbl' and 'data.frame':    406829 obs. of  10 variables:
##  $ InvoiceNo  : num  536365 536365 536365 536365 536365 ...
##  $ StockCode  : chr  "85123A" "71053" "84406B" "84029G" ...
```

```
##  $ Description: Factor w/ 3885 levels "10 COLOUR SPACEBOY PEN",..: 3706 3714 850 1803 2766 2966 1433
##  $ Quantity   : num  6 6 8 6 6 2 6 6 6 32 ...
##  $ InvoiceDate: POSIXct, format: "2010-12-01 08:26:00" "2010-12-01 08:26:00" ...
##  $ UnitPrice  : num  2.55 3.39 2.75 3.39 3.39 7.65 4.25 1.85 1.85 1.69 ...
##  $ CustomerID : num  17850 17850 17850 17850 17850 ...
##  $ Country    : Factor w/ 37 levels "Australia","Austria",..: 35 35 35 35 35 35 35 35 35 35 ...
##  $ Date       : Date, format: "2010-12-01" "2010-12-01" ...
##  $ Time       : chr  "08:26:00" "08:26:00" "08:26:00" "08:26:00" ...
```

After preprocessing, the dataset includes 406,829 records and 10 fields: InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country, Date, Time.

What time do people often purchase online?

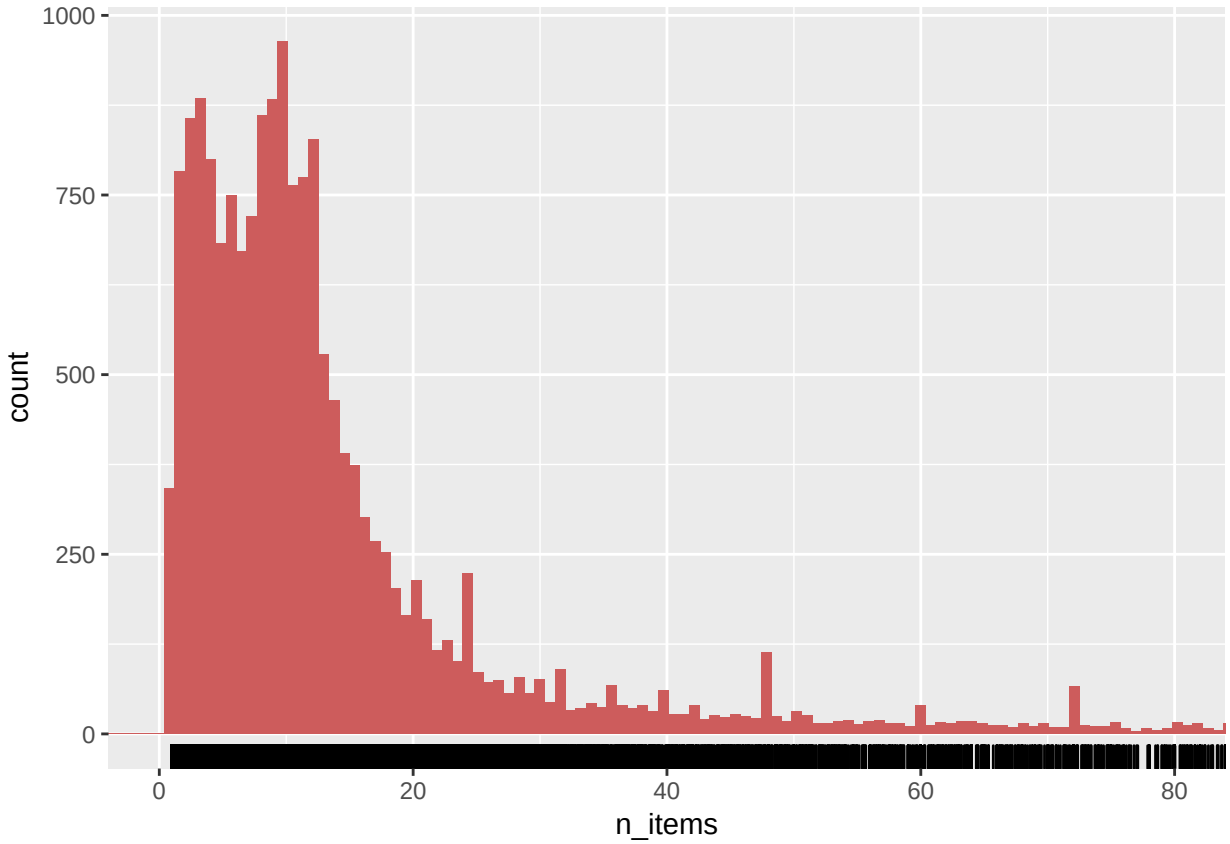
In order to find the answer to this question, we need to extract "hour" from the time column.



There is a clear effect of hour of day on order volume. Most orders happened between 11:00-15:00.

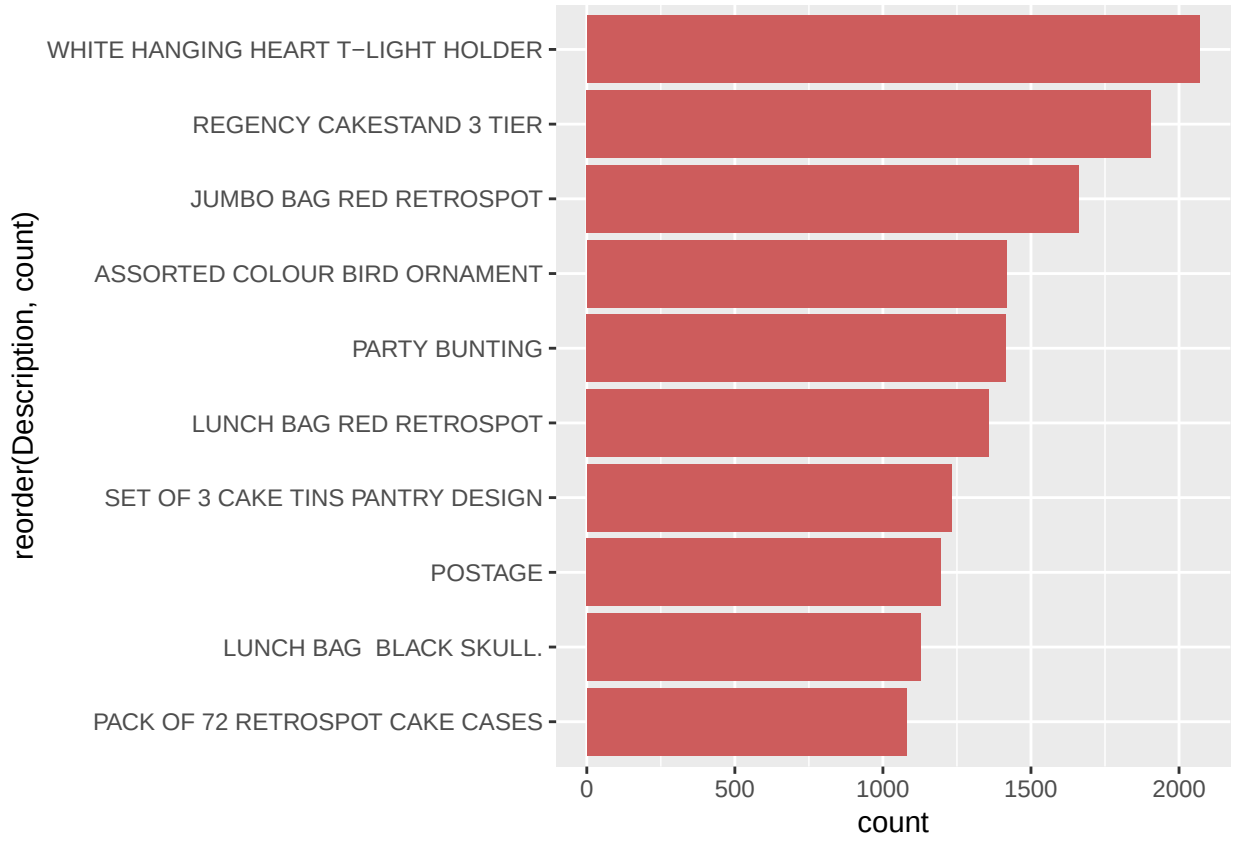
How many items each customer buy?

People mostly purchase less than 10 items (less than 10 items in each invoice). Those negative numbers should be returns.



Top 10 best sellers

```
## # A tibble: 10 x 3
## # Groups:   StockCode [10]
##    StockCode Description                        count
##    <chr>     <fct>                              <int>
##  1 85123A    WHITE HANGING HEART T-LIGHT HOLDER  2070
##  2 22423     REGENCY CAKESTAND 3 TIER            1905
##  3 85099B    JUMBO BAG RED RETROSPOT             1662
##  4 84879     ASSORTED COLOUR BIRD ORNAMENT       1418
##  5 47566     PARTY BUNTING                       1416
##  6 20725     LUNCH BAG RED RETROSPOT             1358
##  7 22720     SET OF 3 CAKE TINS PANTRY DESIGN    1232
##  8 POST      POSTAGE                             1196
##  9 20727     LUNCH BAG  BLACK SKULL.             1126
## 10 21212     PACK OF 72 RETROSPOT CAKE CASES     1080
```



Association rules for online retailer

Before using any rule mining algorithm, we need to transform data from the data frame format into transactions such that we have all the items bought together in one row. For example, this is the format we need:

The function ddply() accepts a data frame, splits it into pieces based on one or more factors, computes on the pieces, then returns the results as a data frame. We use "," to separate different items.

We only need item transactions, so, remove customerID and Date columns.

Write the data from to a csv file and check whether our transaction format is correct.

Perfect! Now we have our transaction dataset shows the matrix of items being bought together. We don't actually see how often they are bought together, we don't see rules either. But we are going to find out.

Let's have a closer look how many transaction we have and what they are.

```
## [1] "Description of the transactions"
```

```
## transactions in sparse format with
##  19297 transactions (rows) and
##  27165 items (columns)
```

```
## transactions as itemMatrix in sparse format with
##  19297 rows (elements/itemsets/transactions) and
##  27165 columns (items) and a density of 0.0006701659
##
## most frequent items:
## WHITE HANGING HEART T-LIGHT HOLDER           REGENCY CAKESTAND 3 TIER
```

```
##                                1758                                1660
##            JUMBO BAG RED RETROSPOT                       PARTY BUNTING
##                                1434                                1271
##      ASSORTED COLOUR BIRD ORNAMENT                             (Other)
##                                1237                              343943
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
##    1 2263 1189  851  768  725  662  618  597  582  554  572  506  487  508
##   16   17   18   19   20   21   22   23   24   25   26   27   28   29   30
##  504  503  449  413  477  420  383  304  313  270  237  253  223  204  222
##   31   32   33   34   35   36   37   38   39   40   41   42   43   44   45
##  216  171  147  138  147  130  111  116   89  104   96   92   85   94   61
##   46   47   48   49   50   51   52   53   54   55   56   57   58   59   60
##   67   73   67   64   52   49   59   50   41   53   50   35   24   40   35
##   61   62   63   64   65   66   67   68   69   70   71   72   73   74   75
##   29   27   23   21   21   17   27   31   24   16   24   18   19   18   13
##   76   77   78   79   80   81   82   83   84   85   86   87   88   89   90
##   14   17   14    7    9   18   17   11   10    8   13   10   14    6    7
##   91   92   93   94   95   96   97   98   99  100  101  102  103  104  105
##    9    6    7    8    5    4    5    5    3    3    3    4    5    5    2
##  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120
##    3    3    7    4    6    3    4    1    2    2    1    3    4    3    1
##  121  122  123  124  126  127  128  132  133  134  135  140  141  142  143
##    2    1    3    2    4    1    1    1    1    3    1    1    1    1    2
##  144  146  147  148  150  151  155  158  162  167  169  172  178  179  181
##    1    1    3    1    1    1    2    2    1    1    1    2    1    1    1
##  199  200  203  205  206  210  230  237  250  251  287  322  402  421
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    5.00   13.00   18.21   24.00  421.00
##
## includes extended item information - examples:
##     labels
## 1        1
## 2 1 HANGER
## 3       10
```

We see 19,296 transactions, this is the number of rows as well, and 7,881 items, remember items are the product descriptions in our original dataset. Transaction here is the collections or subsets of these 7,881 items.

The summary gives us some useful information:

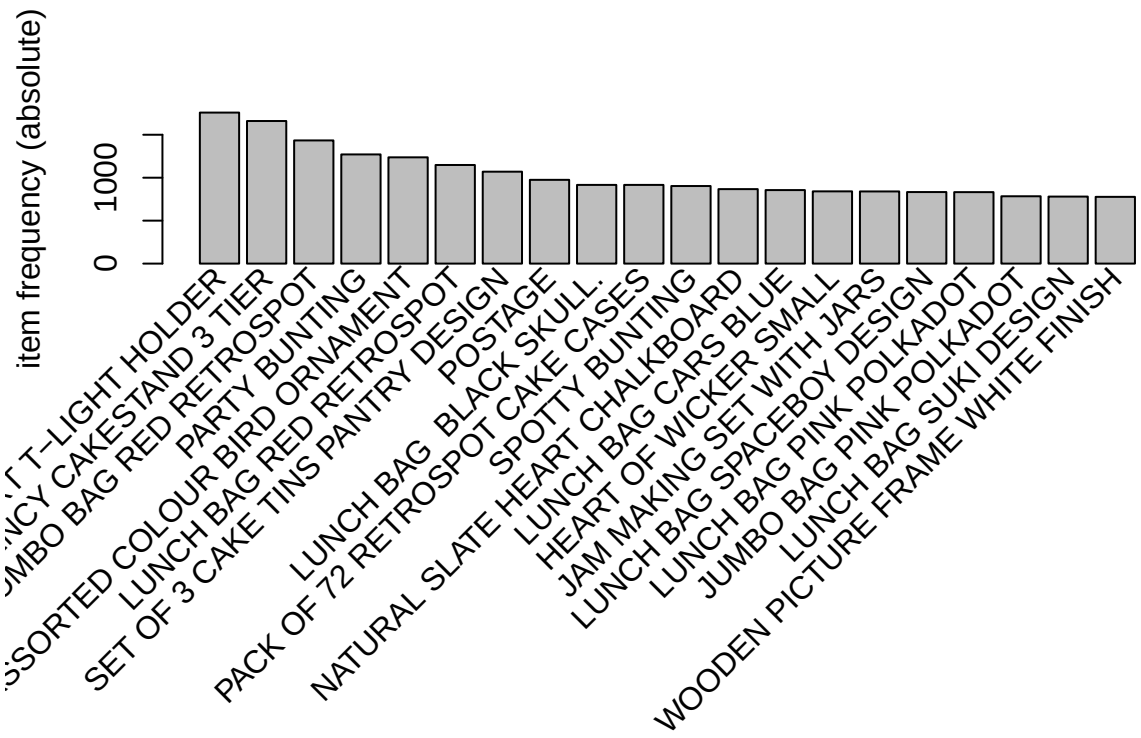
- density: The percentage of non-empty cells in the sparse matrix. In another word, the total number of items that purchased divided by the total number of possible items in that matrix. We can calculate how many items were purchased using density like so:

19296 X 7881 X 0.0022

- The most frequent items should be same with our results in Figure 3.
- For the sizes of the transactions, 2247 transactions for just 1 items, 1147 transactions for 2 items, all the way up to the biggest transaction: 1 transaction for 420 items. This indicates that most customers buy small number of items on each purchase.

- The data distribution is right skewed.

Let's have a look item freqnency plot, this should be in align with Figure 3.



Create some rules

- We use the Apriori algorithm in arules library to mine frequent itemsets and association rules. The algorithm employs level-wise search for frequent itemsets.
- We pass supp=0.001 and conf=0.8 to return all the rules have a support of at least 0.1% and confidence of at least 80%.
- We sort the rules by decreasing confidence.
- Have a look the summary of the rules.

```
## 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 FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
```

```
## Absolute minimum support count: 19
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[27165 item(s), 19297 transaction(s)] done [0.21s].
## sorting and recoding items ... [2407 item(s)] done [0.01s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 5 6 7 8 9 10 done [0.45s].
## writing ... [87110 rule(s)] done [0.05s].
## creating S4 object  ... done [0.06s].
```

```
## set of 87110 rules
##
## rule length distribution (lhs + rhs):sizes
##     2     3     4     5     6     7     8     9    10
##   105  3133  9732 26228 29873 14020  3218   680   121
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   5.000   6.000   5.627   6.000  10.000
##
## summary of quality measures:
##     support           confidence          lift             count
##  Min.   :0.001036   Min.   :0.8000   Min.   :  8.781   Min.   : 20.00
##  1st Qu.:0.001088   1st Qu.:0.8333   1st Qu.: 19.305   1st Qu.: 21.00
##  Median :0.001192   Median :0.8750   Median : 24.786   Median : 23.00
##  Mean   :0.001383   Mean   :0.8834   Mean   : 50.921   Mean   : 26.69
##  3rd Qu.:0.001503   3rd Qu.:0.9231   3rd Qu.: 43.662   3rd Qu.: 29.00
##  Max.   :0.018086   Max.   :1.0000   Max.   :622.484   Max.   :349.00
##
## mining info:
##  data ntransactions support confidence
##    tr         19297   0.001        0.8
```

- The number of rules: 89,697.
- The distribution of rules by length: Most rules are 6 items long.
- The summary of quality measures: ranges of support, confidence, and lift.
- The information on the data mining: total data mined, and minimum parameters we set earlier.

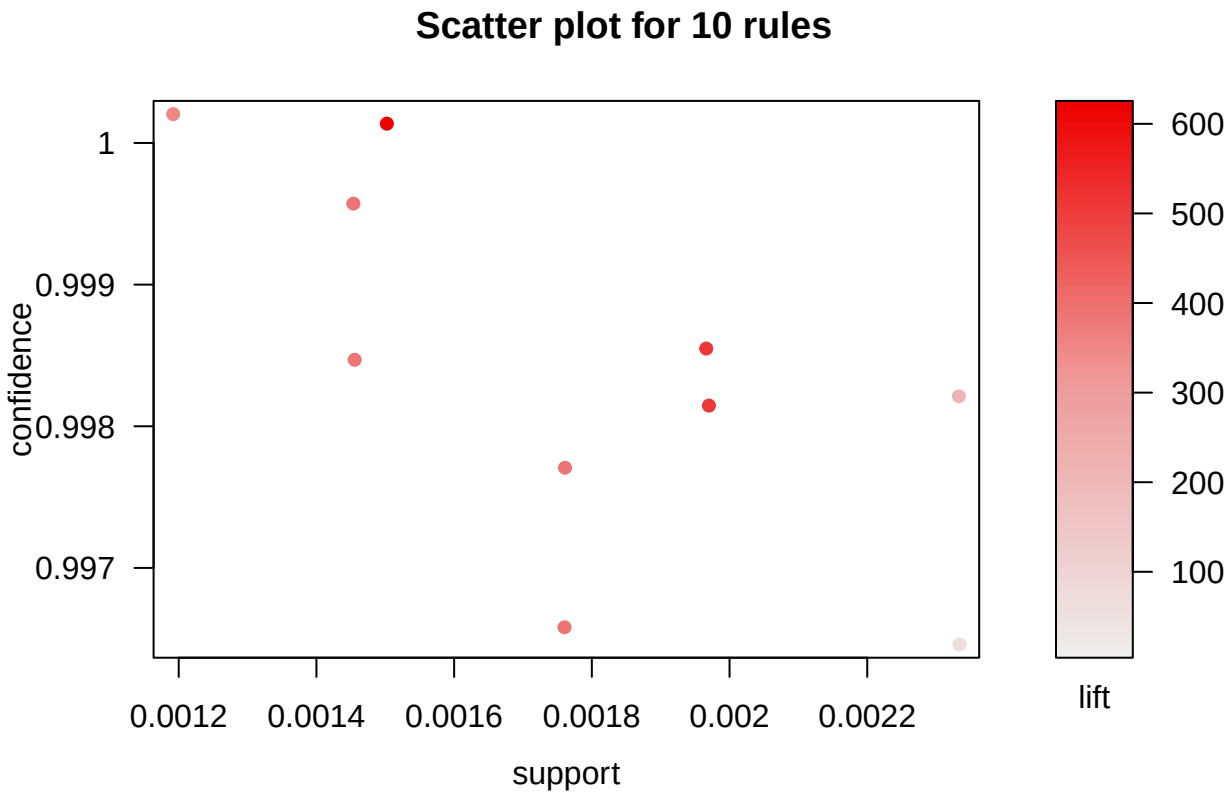
We have 89,697 rules, I don't want to print them all, let's inspect top 10.

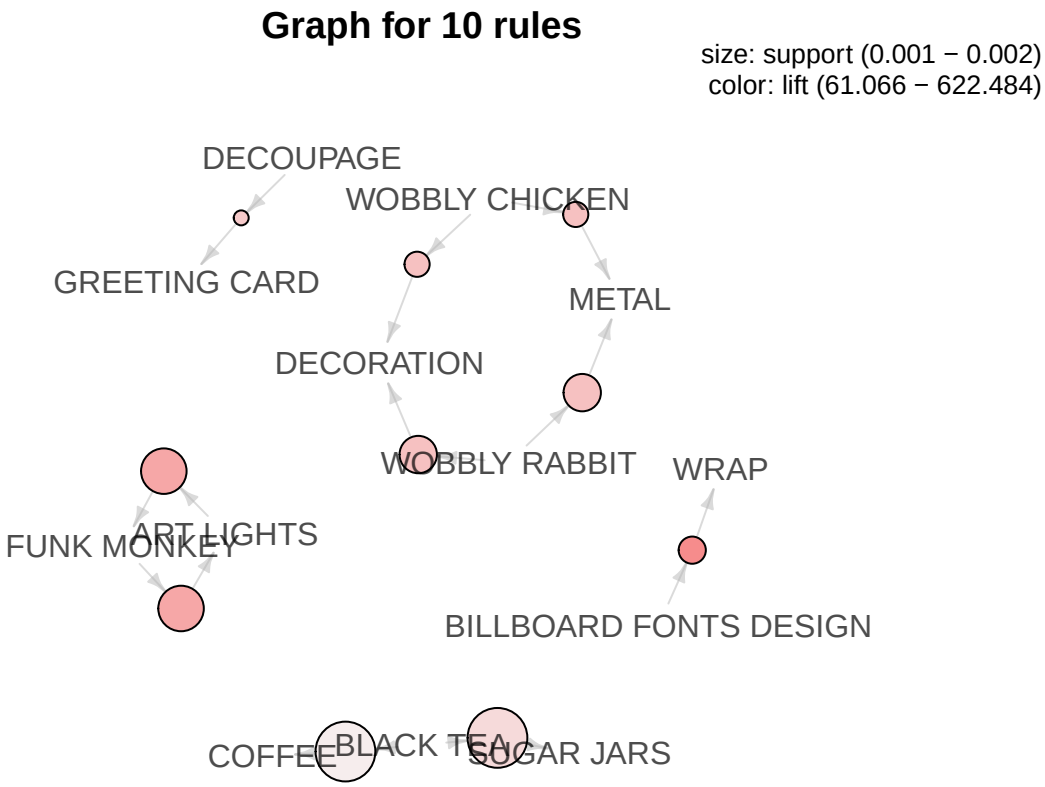
```
##      lhs                          rhs              support     confidence
## [1]  {WOBBLY CHICKEN}          => {DECORATION}     0.001451003 1
## [2]  {WOBBLY CHICKEN}          => {METAL}          0.001451003 1
## [3]  {DECOUPAGE}               => {GREETING CARD}  0.001191895 1
## [4]  {BILLBOARD FONTS DESIGN}  => {WRAP}           0.001502824 1
## [5]  {WOBBLY RABBIT}           => {DECORATION}     0.001761932 1
## [6]  {WOBBLY RABBIT}           => {METAL}          0.001761932 1
## [7]  {BLACK TEA}               => {SUGAR JARS}     0.002331969 1
## [8]  {BLACK TEA}               => {COFFEE}         0.002331969 1
## [9]  {ART LIGHTS}              => {FUNK MONKEY}    0.001969218 1
## [10] {FUNK MONKEY}             => {ART LIGHTS}     0.001969218 1
##      lift      count
## [1]  385.94000 28
## [2]  385.94000 28
## [3]  344.58929 23
## [4]  622.48387 29
## [5]  385.94000 34
```

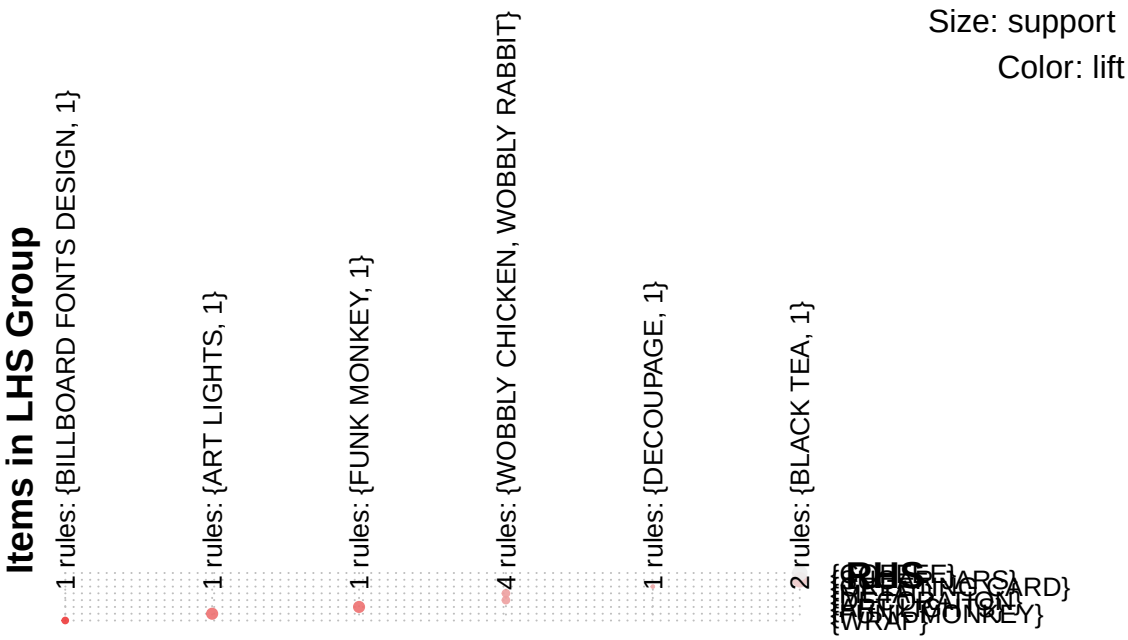
```
## [6]  385.94000 34
## [7]  212.05495 45
## [8]   61.06646 45
## [9]  507.81579 38
## [10] 507.81579 38
```

- 100% customers who bought “WOBBLY CHICKEN” end up bought “DECORATION” as well.
- 100% customers who bought “BLACK TEA” end up bought “SUGAR JAR” as well.

And plot these top 10 rules.




Graph for 10 rules
size: support (0.001 − 0.002)
color: lift (61.066 − 622.484)
DECOUPAGE
WOBBLY CHICKEN
GREETING CARD
METAL
DECORATION
WOBBLY RABBIT
WRAP
ART LIGHTS
FUNK MONKEY
BILLBOARD FONTS DESIGN
COFFEE
BLACK TEA
SUGAR JARS



reference: R and Data Mining