Association Rules

Basket Market Analysis

Dante Conti, Sergi Ramirez, (c) IDEAI

2025-10-20

Table of contents

0.1 Almacenamiento de datos de transacciones

0.1.1 Formato de matriz binaria

Cada fila representa una transacción y cada columna representa uno de los posibles ítems que podrían formar parte de ella. Las celdas de la matriz se rellenan con valores binarios (1 o 0) que indican si un artículo está presente (1) o ausente (0) en una transacción concreta.

Este es el formato de entrada que utilizan la mayoría de algoritmos, sin embargo, no suele ser el más adecuado para almacenar las transacciones ya que requiere conocer de antemano todos los posibles ítems para crear las columnas y, dado que cada transacción suele contener solo una pequeña proporción de los posibles ítems, la mayoría de valores son cero (sparse).

0.1.2 Formato de lista *(basket)*

Cada transacción se representa como una lista de los ítems que forman parte de ella y se le asigna un identificador único por transacción. Este formato es más eficiente desde el punto de vista de la memoria, ya que sólo almacena los elementos presentes en cada transacción.

0.2 R

```
## ----libs-----------------------------------------------------------------
# install.packages("devtools")
# devtools::install_github("rsquaredacademy/mbar")
library(readxl)
library(readr)
library(mbar)
library(arules)
library(arulesViz)
library(magrittr)
library(dplyr)
library(lubridate)
library(forcats)
library(ggplot2)
```

```r
## ----preprocess----------------------------------------------------------
mba_data <- read_excel("online-retail.xlsx")
transactions <- mbar_prep_data(mba_data, InvoiceNo, Description)
head(transactions)
```

```
# A tibble: 6 x 1,114
  item_1 item_2 item_3 item_4 item_5 item_6 item_7 item_8 item_9 item_10 item_11
  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>   <chr>
1 WHITE~ "WHIT~ "CREA~ "KNIT~ "RED ~ "SET ~ "GLAS~ ""     ""     ""      ""
2 HAND ~ "HAND~ ""     ""     ""     ""     ""     ""     ""     ""      ""
3 ASSOR~ "POPP~ "POPP~ "FELT~ "IVOR~ "BOX ~ "BOX ~ "BOX ~ "HOME~ "LOVE ~ "RECIP~
4 JAM M~ "RED ~ "YELL~ "BLUE~ ""     ""     ""     ""     ""     ""      ""
5 BATH ~ ""     ""     ""     ""     ""     ""     ""     ""     ""      ""
6 ALARM~ "ALAR~ "ALAR~ "PAND~ "STAR~ "INFL~ "VINT~ "SET/~ "ROUN~ "SPACE~ "LUNCH~
# i 1,103 more variables: item_12 <chr>, item_13 <chr>, item_14 <chr>,
#   item_15 <chr>, item_16 <chr>, item_17 <chr>, item_18 <chr>, item_19 <chr>,
#   item_20 <chr>, item_21 <chr>, item_22 <chr>, item_23 <chr>, item_24 <chr>,
#   item_25 <chr>, item_26 <chr>, item_27 <chr>, item_28 <chr>, item_29 <chr>,
#   item_30 <chr>, item_31 <chr>, item_32 <chr>, item_33 <chr>, item_34 <chr>,
#   item_35 <chr>, item_36 <chr>, item_37 <chr>, item_38 <chr>, item_39 <chr>,
#   item_40 <chr>, item_41 <chr>, item_42 <chr>, item_43 <chr>, ...
```

```r
## ----read_data-----------------------------------------------------------
(basket_data <- read.transactions("transaction_data.csv", format = "basket",
  sep = ","))
```

```
transactions in sparse format with
 25901 transactions (rows) and
 10085 items (columns)
```

```r
## ----summary-------------------------------------------------------------
summary(basket_data)
```

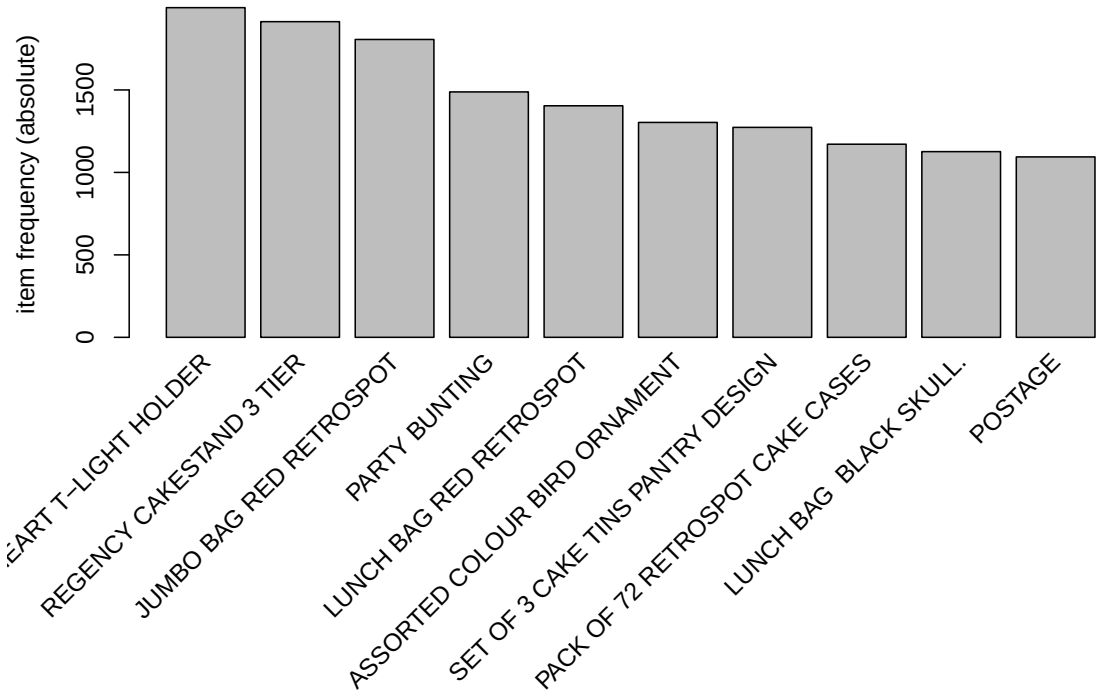
```
transactions as itemMatrix in sparse format with
 25901 rows (elements/itemsets/transactions) and
 10085 columns (items) and a density of 0.001660018

most frequent items:
WHITE HANGING HEART T-LIGHT HOLDER           REGENCY CAKESTAND 3 TIER
                              1999                               1914
```

```
           JUMBO BAG RED RETROSPOT                       PARTY BUNTING
                              1806                                1488
           LUNCH BAG RED RETROSPOT                             (Other)
                              1404                              425005

element (itemset/transaction) length distribution:
sizes
   0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
1454 4578 1727 1208  942  891  781  715  696  683  612  642  547  530  543  555
  16   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31
 537  479  459  491  428  405  328  311  280  248  261  235  221  233  224  175
  32   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47
 174  145  149  139  122  119  100  117   98   94  102   93   72   73   74   71
  48   49   50   51   52   53   54   55   56   57   58   59   60   61   62   63
  69   68   59   70   49   49   54   57   42   32   42   39   34   40   22   27
  64   65   66   67   68   69   70   71   72   73   74   75   76   77   78   79
  30   24   34   28   25   21   23   26   14   17   24   11   18   14   13   10
  80   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95
  16   18   15   10    9   16   13   16   13    7    8   12   12    8    7    7
  96   97   98   99  100  101  102  103  104  105  106  107  108  109  110  111
   4    7    9    5    8    8    4    5    7    2    3    7    9    4    7    4
 112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127
   2    7    1    1    4    7    6    2    3    5    4    4    2    5    6    2
 129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144
   1    4    3    6    6    3    4    3    2    1    1    3    8    5    3    4
 145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160
   4    6    2    3    1    4    3    2    4    7    3    3    5    2    4    5
 162  163  164  167  168  169  170  171  172  173  174  175  176  177  178  179
   1    2    1    3    5    2    2    4    3    1    3    5    1    2    2    2
 180  181  182  183  184  185  186  187  189  190  192  193  194  196  197  198
   2    1    2    1    2    1    1    2    2    1    1    5    1    2    3    2
 201  202  204  205  206  207  208  209  212  213  215  219  220  224  226  227
   1    1    2    2    1    3    3    2    1    2    2    7    1    3    3    1
 228  230  232  234  236  238  240  241  244  248  249  250  252  256  257  258
   1    2    1    2    1    2    2    2    1    1    2    2    1    1    1    1
 260  261  263  265  266  270  272  281  284  285  298  299  301  303  304  305
   2    1    2    1    1    1    1    1    1    2    1    2    1    1    1    3
 312  314  316  320  321  326  327  329  332  333  338  339  341  344  348  350
   2    1    1    2    1    1    1    1    1    1    1    1    1    2    1    1
 360  365  367  375  391  394  398  400  402  405  411  419  422  429  431  442
   2    1    1    3    1    1    1    1    1    1    1    2    1    1    2    1
 447  460  468  471  477  509  514  530  587  627 1114
   1    1    1    1    1    1    1    1    1    1    1
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    2.00    8.00   16.74   20.00 1114.00

includes extended item information - examples:
                   labels
1   *Boombox Ipod Classic
2 *USB Office Mirror Ball
3                        ?
```

```
## ----format = single-------------------------------------------------------
get_data <- read.transactions("retail.csv",
  format = "single",
  sep = ",",
  cols = c("InvoiceNo", "item"))
```

```
## ----item frequency plot---------------------------------------------------
itemFrequencyPlot(basket_data, topN = 10, type = 'absolute')
```



0.3 Python

0.4 Mlxtend

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder

dataset = [['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
           ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
           ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
           ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
           ['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']]

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
```

0.5 Arules

0.6 Generación de les reglas de asociación a partir de conjuntos de elementos freceuntes

0.6.1 A Priori

Propuesto por Agrawal and Srikant en 1994 es uno de los primeros algoritmos desarrollados para la identificación de itemsets frecuentes en bases de datos con transacciones, y posterior conversión a reglas de asociación. Si bien sucesivas modificaciones del algoritmo original han conseguido mejoras notables en su rendimiento, sigue siendo muy empleado. Este algoritmo tiene dos etapas:

- Identificar todos los itemsets que ocurren con una frecuencia por encima de un determinado límite (itemsets frecuentes).
- Convertir los itemsets frecuentes en reglas de asociación.

0.7 R

```
## ----generate rules-----------------------------------------------------
rules <- apriori(basket_data, parameter = list(supp=0.009, conf=0.8,
    target = "rules", maxlen = 4))
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.8    0.1    1 none FALSE            TRUE       5   0.009      1
 maxlen target  ext
      4  rules TRUE

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

Absolute minimum support count: 233

set item appearances ...[0 item(s)] done [0.00s].
```

```
set transactions ...[10085 item(s), 25901 transaction(s)] done [0.06s].
sorting and recoding items ... [508 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4


 done [0.01s].
writing ... [22 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

```
## ----rules summary-----------------------------------------------------
summary(rules)
```

```
set of 22 rules

rule length distribution (lhs + rhs):sizes
 2  3  4
11  9  2

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   2.000   2.500   2.591   3.000   4.000

summary of quality measures:
    support           confidence        coverage             lift
 Min.   :0.009034   Min.   :0.8035   Min.   :0.009614   Min.   :22.59
 1st Qu.:0.010453   1st Qu.:0.8530   1st Qu.:0.011592   1st Qu.:25.02
 Median :0.013223   Median :0.8868   Median :0.014362   Median :55.94
 Mean   :0.012760   Mean   :0.9120   Mean   :0.014061   Mean   :48.55
 3rd Qu.:0.014362   3rd Qu.:1.0000   3rd Qu.:0.014362   3rd Qu.:61.23
 Max.   :0.018339   Max.   :1.0000   Max.   :0.021544   Max.   :71.30
     count
 Min.   :234.0
 1st Qu.:270.8
 Median :342.5
 Mean   :330.5
 3rd Qu.:372.0
 Max.   :475.0

mining info:
        data ntransactions support confidence
 basket_data         25901   0.009        0.8

 apriori(data = basket_data, parameter = list(supp = 0.009, conf = 0.8, target = "rules", ma
```

0.8 Python

```
from mlxtend.frequent_patterns import apriori
frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)
frequent_itemsets
```

```
    support                    itemsets
0       0.8                      (Eggs)
1       1.0               (Kidney Beans)
2       0.6                      (Milk)
3       0.6                     (Onion)
4       0.6                    (Yogurt)
5       0.8         (Eggs, Kidney Beans)
6       0.6                (Onion, Eggs)
7       0.6         (Milk, Kidney Beans)
8       0.6        (Onion, Kidney Beans)
9       0.6       (Kidney Beans, Yogurt)
10      0.6  (Onion, Eggs, Kidney Beans)
```

0.8.1 Eclat

En el 2000, Zaki propuso un nuevo algoritmo para encontrar patrones frecuentes (itemsets frecuentes) llamado Equivalence Class Transformation (Eclat). La principal diferencia entre este algoritmo y el Apriori es la forma en que se escanean y analizan los datos. El algoritmo Apriori emplea transacciones almacenadas de forma horizontal (lista o basket), es decir, todos los elementos que forman una misma transacción están en la misma línea. El algoritmo Eclat, sin embargo, analiza las transacciones en formato vertical, donde cada línea contiene un único ítem y el identificador de las transacciones en las que aparece ese ítem.

0.9 R

```
rules_ECLAT <- eclat(basket_data, parameter = list(supp=0.009, maxlen = 4))
```

```
Eclat

parameter specification:
 tidLists support minlen maxlen            target  ext
    FALSE   0.009      1      4 frequent itemsets TRUE
```

```
algorithmic control:
 sparse sort verbose
      7   -2    TRUE

Absolute minimum support count: 233

create itemset ...
set transactions ...[10085 item(s), 25901 transaction(s)] done [0.06s].
sorting and recoding items ... [508 item(s)] done [0.00s].
creating sparse bit matrix ... [508 row(s), 25901 column(s)] done [0.00s].
writing  ... [742 set(s)] done [0.21s].
Creating S4 object  ... done [0.00s].
```

```
inspect(sort(rules_ECLAT, by = 'support')[1:10])
```

```
     items                                 support    count
[1]  {WHITE HANGING HEART T-LIGHT HOLDER} 0.07717849 1999
[2]  {REGENCY CAKESTAND 3 TIER}           0.07389676 1914
[3]  {JUMBO BAG RED RETROSPOT}            0.06972704 1806
[4]  {PARTY BUNTING}                      0.05744952 1488
[5]  {LUNCH BAG RED RETROSPOT}            0.05420640 1404
[6]  {ASSORTED COLOUR BIRD ORNAMENT}      0.05030694 1303
[7]  {SET OF 3 CAKE TINS PANTRY DESIGN}   0.04914868 1273
[8]  {PACK OF 72 RETROSPOT CAKE CASES}    0.04521061 1171
[9]  {LUNCH BAG  BLACK SKULL.}            0.04347322 1126
[10] {POSTAGE}                            0.04223775 1094
```

0.10 Python

```
from mlxtend.frequent_patterns import fpmax
frequent_itemsets = fpmax(df, min_support=0.6, use_colnames=True)
frequent_itemsets
```

```
   support                    itemsets
0      0.6  (Onion, Eggs, Kidney Beans)
1      0.6         (Milk, Kidney Beans)
2      0.6       (Kidney Beans, Yogurt)
```

0.10.1 FP-Growth

En el 2000, Zaki propuso un nuevo algoritmo para encontrar patrones frecuentes (itemsets frecuentes) llamado Equivalence Class Transformation (Eclat). La principal diferencia entre este algoritmo y el Apriori es la forma en que se escanean y analizan los datos. El algoritmo Apriori emplea transacciones almacenadas de forma horizontal (lista o basket), es decir, todos los elementos que forman una misma transacción están en la misma línea. El algoritmo Eclat, sin embargo, analiza las transacciones en formato vertical, donde cada línea contiene un único ítem y el identificador de las transacciones en las que aparece ese ítem.

0.11 R

0.12 Python

```
from mlxtend.frequent_patterns import fpgrowth
frequent_itemsets = fpgrowth(df, min_support=0.6, use_colnames=True)
frequent_itemsets
```

```
    support                      itemsets
0       1.0                 (Kidney Beans)
1       0.8                         (Eggs)
2       0.6                       (Yogurt)
3       0.6                         (Milk)
4       0.6                        (Onion)
5       0.8           (Eggs, Kidney Beans)
6       0.6         (Kidney Beans, Yogurt)
7       0.6           (Milk, Kidney Beans)
8       0.6                  (Onion, Eggs)
9       0.6          (Onion, Kidney Beans)
10      0.6  (Onion, Eggs, Kidney Beans)
```

0.13 Generación de reglas y criterios de selección

0.13.1 Soporte *(Support)*

El soporte es una métrica asociada al conjunto de *items* y no a las reglas de asociación en si. Existen los siguientes tipos de soporte:

- **soporte antecedente**: Proporción de transacciones que contienen el antecedente (A)
- **soporte consecuente**: Proporción de transacciones que contienen el consecuente (C)

- **soporte**: Proporción de transacciones que combinan el antecedente (A) y el consecuente (C)

El soporte se utiliza para medir la abundancia o frecuencia (significancia o importancia) de un *item sets* en una base de datos.

$Support(A \to C) = support(A \cup C)$

El rango del soporte va entre 0 y 1.

0.13.2 Confianza *(Confidence)*

La confianza de una regla $A \to C$ es la probabilidad de ver el consecuente en una transacción, dado que también contiene el antecedente. La regla no es simétrica y su valor máximo es 1 que significa que el consecuente y el antecedente siempre ocurren juntos y el mínimo es 0.

$confidence(A \to C) = \frac{support(A \to C)}{support(A)}$

0.13.3 Elevación *(Lift)*

La métrica de elevación se utiliza comúnmente para medir con qué frecuencia el antecedente y el consecuente de una regla $A \to C$ ocurren juntos con mayor frecuencia de lo que esperaríamos si fueran estadísticamente independientes. Si A y C son independientes, la puntuación de elevación será exactamente 1.

$lift(A \to C) = \frac{confidence(A \to C)}{support(C)}, range : [0, \infty)$

0.13.4 Apalancamiento *(Leverage)*

El apalancamiento calcula la diferencia entre la frecuencia observada de A y C apareciendo juntos y la frecuencia esperada si A y C fueran independientes. Un valor de apalancamiento de 0 indica independencia.

$leverage(A \to C) = support(A \to C) - support(A) \cdot support(C), range : [-1, 1]$

0.13.5 Conviction

Un valor alto de convicción significa que el consecuente depende en gran medida del antecedente. Por ejemplo, en el caso de una puntuación de confianza perfecta, el denominador se convierte en 0 (debido a 1 - 1), por lo que la puntuación de convicción se define como 'inf'. Similar al caso de la elevación, si los elementos son independientes, la convicción es 1.

$Conviction(A \to C) = \frac{1 - support(C)}{1 - confidence(A \to C)}, range : [0, \infty)$

0.13.6 Zhangs Metric

Mide tanto la asociación como la disociación. El valor oscila entre -1 y 1. Un valor positivo (> 0) indica asociación y un valor negativo, disociación.

$ZhangMetric(A \to C) = \frac{confidence(A \to C) - confidence(A' \to C)}{max(confidence(A \to C), confidence(A' \to C))}, range : [-1, 1]$

0.14 R

```
## ----inspect rules-------------------------------------------------------
basket_rules <- sort(rules, by = 'confidence', decreasing = TRUE)
inspect(basket_rules[1:10])
```

```
     lhs                                   rhs                                 support confi
[1]  {BACK DOOR}                        => {KEY FOB}                         0.009613528  1.00
[2]  {SET 3 RETROSPOT TEA}              => {SUGAR}                           0.014362380  1.00
[3]  {SUGAR}                            => {SET 3 RETROSPOT TEA}             0.014362380  1.00
[4]  {SET 3 RETROSPOT TEA}              => {COFFEE}                          0.014362380  1.00
[5]  {SUGAR}                            => {COFFEE}                          0.014362380  1.00
[6]  {SHED}                             => {KEY FOB}                         0.011273696  1.00
[7]  {SET 3 RETROSPOT TEA,
      SUGAR}                            => {COFFEE}                          0.014362380  1.00
[8]  {COFFEE,
      SET 3 RETROSPOT TEA}              => {SUGAR}                           0.014362380  1.00
[9]  {COFFEE,
      SUGAR}                            => {SET 3 RETROSPOT TEA}             0.014362380  1.00
[10] {PINK REGENCY TEACUP AND SAUCER,
      REGENCY CAKESTAND 3 TIER,
      ROSES REGENCY TEACUP AND SAUCER} => {GREEN REGENCY TEACUP AND SAUCER} 0.009999614  0.89
```

```
## ----top rules by support-----------------------------------------------
supp_rules <- sort(rules, by = 'support', decreasing = TRUE)
top_rules <- supp_rules[1:10]
inspect(top_rules)
```

```
     lhs                                    rhs                                 support conf
[1]  {PINK REGENCY TEACUP AND SAUCER,
      ROSES REGENCY TEACUP AND SAUCER}  => {GREEN REGENCY TEACUP AND SAUCER} 0.01833906  0.88
[2]  {GREEN REGENCY TEACUP AND SAUCER,
      PINK REGENCY TEACUP AND SAUCER}   => {ROSES REGENCY TEACUP AND SAUCER} 0.01833906  0.85
```

```
[3]  {SET 3 RETROSPOT TEA}              => {SUGAR}                          0.01436238  1.00
[4]  {SUGAR}                            => {SET 3 RETROSPOT TEA}            0.01436238  1.00
[5]  {SET 3 RETROSPOT TEA}              => {COFFEE}                         0.01436238  1.00
[6]  {COFFEE}                           => {SET 3 RETROSPOT TEA}            0.01436238  0.80
[7]  {SUGAR}                            => {COFFEE}                         0.01436238  1.00
[8]  {COFFEE}                           => {SUGAR}                          0.01436238  0.80
[9]  {SET 3 RETROSPOT TEA,
      SUGAR}                            => {COFFEE}                         0.01436238  1.00
[10] {COFFEE,
      SET 3 RETROSPOT TEA}              => {SUGAR}                          0.01436238  1.00
```

```
## ----top rules by confidence-----------------------------------------------
supp_rules <- sort(rules, by = 'confidence', decreasing = TRUE)
top_rules <- supp_rules[1:10]
inspect(top_rules)
```

```
     lhs                                   rhs                                    support confi
[1]  {BACK DOOR}                        => {KEY FOB}                           0.009613528  1.00
[2]  {SET 3 RETROSPOT TEA}              => {SUGAR}                             0.014362380  1.00
[3]  {SUGAR}                            => {SET 3 RETROSPOT TEA}               0.014362380  1.00
[4]  {SET 3 RETROSPOT TEA}              => {COFFEE}                            0.014362380  1.00
[5]  {SUGAR}                            => {COFFEE}                            0.014362380  1.00
[6]  {SHED}                             => {KEY FOB}                           0.011273696  1.00
[7]  {SET 3 RETROSPOT TEA,
      SUGAR}                            => {COFFEE}                            0.014362380  1.00
[8]  {COFFEE,
      SET 3 RETROSPOT TEA}              => {SUGAR}                             0.014362380  1.00
[9]  {COFFEE,
      SUGAR}                            => {SET 3 RETROSPOT TEA}               0.014362380  1.00
[10] {PINK REGENCY TEACUP AND SAUCER,
      REGENCY CAKESTAND 3 TIER,
      ROSES REGENCY TEACUP AND SAUCER} => {GREEN REGENCY TEACUP AND SAUCER} 0.009999614  0.89
```

```
## ----top rules by lift-----------------------------------------------------
supp_rules <- sort(rules, by = 'lift', decreasing = TRUE)
top_rules <- supp_rules[1:10]
inspect(top_rules)
```

```
     lhs                              rhs                          support
[1]  {REGENCY TEA PLATE PINK}      => {REGENCY TEA PLATE GREEN} 0.009034400
[2]  {SET 3 RETROSPOT TEA}         => {SUGAR}                   0.014362380
```

```
[3]  {SUGAR}                        => {SET 3 RETROSPOT TEA}     0.014362380
[4]  {COFFEE, SET 3 RETROSPOT TEA} => {SUGAR}                    0.014362380
[5]  {COFFEE, SUGAR}                => {SET 3 RETROSPOT TEA}     0.014362380
[6]  {BACK DOOR}                    => {KEY FOB}                 0.009613528
[7]  {SHED}                         => {KEY FOB}                 0.011273696
[8]  {REGENCY TEA PLATE GREEN}      => {REGENCY TEA PLATE ROSES} 0.010347091
[9]  {SET 3 RETROSPOT TEA}          => {COFFEE}                  0.014362380
[10] {COFFEE}                       => {SET 3 RETROSPOT TEA}     0.014362380
     confidence coverage    lift     count
[1]  0.8863636  0.010192657 71.29722 234
[2]  1.0000000  0.014362380 69.62634 372
[3]  1.0000000  0.014362380 69.62634 372
[4]  1.0000000  0.014362380 69.62634 372
[5]  1.0000000  0.014362380 69.62634 372
[6]  1.0000000  0.009613528 61.23168 249
[7]  1.0000000  0.011273696 61.23168 292
[8]  0.8322981  0.012431952 55.99313 268
[9]  1.0000000  0.014362380 55.94168 372
[10] 0.8034557  0.017875758 55.94168 372
```

0.15 Python

```python
from mlxtend.frequent_patterns import association_rules

association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
```

```
              antecedents            consequents  ...  certainty  kulczynski
0                  (Eggs)         (Kidney Beans)  ...      0.000       0.900
1           (Kidney Beans)                 (Eggs)  ...      0.000       0.900
2                 (Yogurt)         (Kidney Beans)  ...      0.000       0.800
3                   (Milk)         (Kidney Beans)  ...      0.000       0.800
4                  (Onion)                 (Eggs)  ...      1.000       0.875
5                  (Eggs)                (Onion)  ...      0.375       0.875
6                  (Onion)         (Kidney Beans)  ...      0.000       0.800
7            (Onion, Eggs)         (Kidney Beans)  ...      0.000       0.800
8   (Onion, Kidney Beans)                 (Eggs)  ...      1.000       0.875
9    (Eggs, Kidney Beans)                (Onion)  ...      0.375       0.875
10                 (Onion)   (Eggs, Kidney Beans)  ...      1.000       0.875
11                 (Eggs)  (Onion, Kidney Beans)  ...      0.375       0.875

[12 rows x 14 columns]
```

0.16 Poda de las reglas de asociación

```
## ----what influenced purchase of sugar-----------------------------------
sugar_rules <- apriori(basket_data, parameter = list(supp = 0.009, conf = 0.8),
  appearance = list(default = "lhs", rhs = "SUGAR"))
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.8    0.1    1 none FALSE            TRUE       5   0.009      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: 233

set item appearances ...[1 item(s)] done [0.00s].
set transactions ...[10085 item(s), 25901 transaction(s)] done [0.06s].
sorting and recoding items ... [508 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.01s].
writing ... [3 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

```
rules_sugar <- sort(sugar_rules, by = "confidence", decreasing = TRUE)
inspect(rules_sugar)
```

```
    lhs                               rhs     support    confidence coverage
[1] {SET 3 RETROSPOT TEA}          => {SUGAR} 0.01436238 1.0000000  0.01436238
[2] {COFFEE, SET 3 RETROSPOT TEA} => {SUGAR} 0.01436238 1.0000000  0.01436238
[3] {COFFEE}                       => {SUGAR} 0.01436238 0.8034557  0.01787576
    lift     count
[1] 69.62634 372
[2] 69.62634 372
[3] 55.94168 372
```

```r
## ----what purchases did sugar influence----------------------------------
sugar_rules <- apriori(basket_data, parameter = list(supp = 0.009, conf = 0.8),
  appearance = list(default = "rhs", lhs = "SUGAR"))
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.8    0.1    1 none FALSE            TRUE       5   0.009      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: 233

set item appearances ...[1 item(s)] done [0.00s].
set transactions ...[10085 item(s), 25901 transaction(s)] done [0.06s].
sorting and recoding items ... [508 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 done [0.00s].
writing ... [2 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

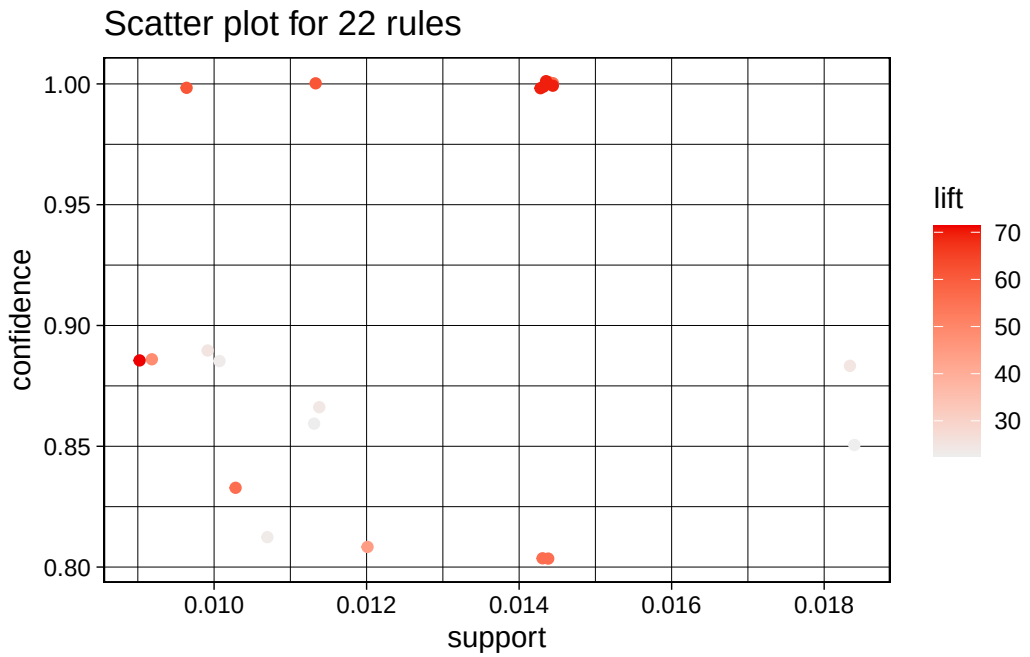
```r
rules_sugar <- sort(sugar_rules, by = "confidence", decreasing = TRUE)
inspect(rules_sugar)
```

```
    lhs        rhs                    support    confidence coverage   lift
[1] {SUGAR} => {SET 3 RETROSPOT TEA} 0.01436238 1          0.01436238 69.62634
[2] {SUGAR} => {COFFEE}              0.01436238 1          0.01436238 55.94168
    count
[1] 372
[2] 372
```

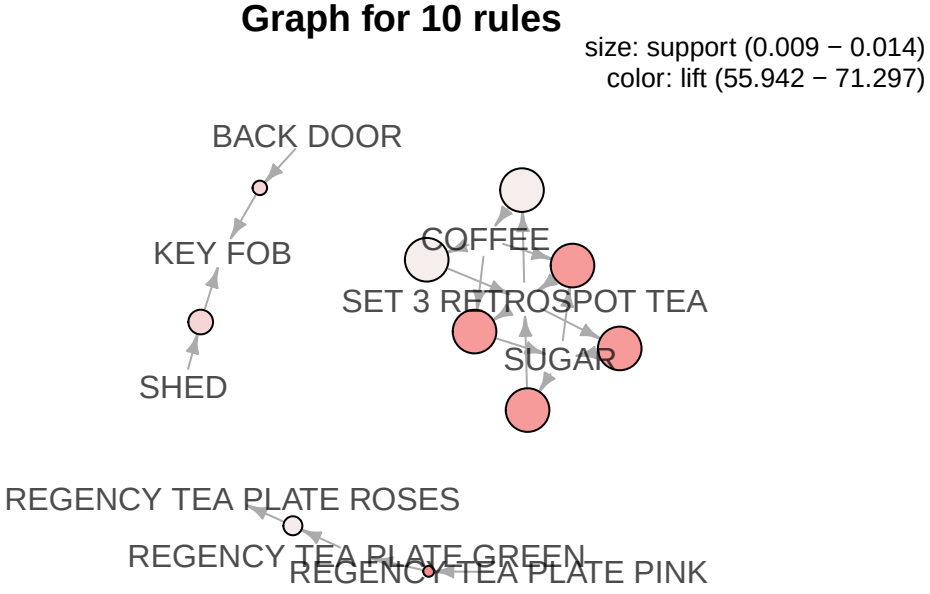
0.17 Visualización de reglas

0.18 R

```
## ----plot rules-----------------------------------------------------------
plot(basket_rules)
```




```
## ----plot rules-----------------------------------------------------------
plot(top_rules, method = "graph", engine = "igraph")
```


Graph for 10 rules
size: support (0.009 − 0.014)
color: lift (55.942 − 71.297)
BACK DOOR
KEY FOB
SHED
COFFEE
SET 3 RETROSPOT TEA
SUGAR
REGENCY TEA PLATE ROSES
REGENCY TEA PLATE GREEN
REGENCY TEA PLATE PINK