

# Métodos Quantitativos

*Prof. Dr. A. L. Korzenowski*

## Aula 06: Market Basket Analysis using R: An Association Rules application

Nosso objetivo é aprender sobre a análise da cesta de mercado e o algoritmo APRIORI que funciona por trás disso. Você verá como isso está ajudando os varejistas a impulsionar os negócios, prevendo quais itens os clientes compram juntos.

Você é um cientista de dados (ou está se tornando um!) e obtém um cliente que administra uma loja de varejo. Seu cliente fornece dados para todas as transações que consistem em itens comprados na loja por vários clientes durante um período de tempo e solicita que você os utilize para ajudar a impulsionar seus negócios. Seu cliente usará suas descobertas para não apenas alterar/atualizar/adicionar itens no inventário, mas também para alterar o layout da loja física ou melhor, uma loja online. Para encontrar resultados que ajudarão seu cliente, você usará o MBA (Market Basket Analysis), que usa a Mineração de Regras de Associação nos dados de transação fornecidos.

### Association Rule Mining

Association Rule Mining é usado quando você deseja encontrar uma associação entre diferentes objetos em um conjunto, encontrar padrões frequentes em um banco de dados de transações, bancos de dados relacionais ou qualquer outro repositório de informações. As aplicações da Association Rule Mining são encontradas em Marketing, Análise de dados de cesta (ou Análise de cesta de mercado) em varejo, clustering e classificação. Ele pode dizer quais itens os clientes costumam comprar juntos, gerando um conjunto de regras chamado Regras de Associação. Em palavras simples, ele fornece a saída como regras em forma, se é isso ou aquilo. Os clientes podem usar essas regras para diversas estratégias de marketing:

- Alterando o layout da loja de acordo com as tendências
- Análise de comportamento do cliente
- Design do catálogo
- Marketing cruzado em lojas online
- Quais são os itens de tendência que os clientes compram
- E-mails personalizados com vendas adicionais

Considere o seguinte exemplo de transações em um mercado:

1. {pão, leite}
2. {pão, fralda, cerveja, ovos}
3. {leite, fralda, cerveja, cola}
4. {pão, leite, cerveja, fralda}
5. {pão, leite, fralda, cola}

Dado é um conjunto de dados de transação. Você pode ver as transações numeradas de 1 a 5. Cada transação mostra itens comprados nessa transação. Você pode ver que a fralda é comprada com cerveja em três transações. Da mesma forma, o pão é comprado com leite em três transações, tornando-os dois conjuntos de itens frequentes. As regras de associação são fornecidas no formulário abaixo:

$A \Rightarrow B$  [Support, Confidence]

A parte antes de  $\Rightarrow$  é referido como *if (antecedente)* e a parte depois de  $\Rightarrow$  é referida como *then (consequente)*, onde  $A$  e  $B$  são conjuntos de itens nos dados de transações.  $A$  e  $B$  são eventos (conjuntos) disjuntos. Veja o seguinte exemplo:

$Computer \Rightarrow Anti-virus\_Softwares$  [Support = 20%, confidence = 60%]

Esta regra diz o seguinte:

1. 20% das transações mostram Anti-vírus software sendo comprado junto com um computador
2. 60% dos compradores que compram Anti-virus Software, o compram junto a uma compra de um computer.

Vamos entender os conceitos básicos de regras de associação.

### Conceitos e definições

1. Itemset: coleção de um ou mais itens. K-item-set significa um conjunto de k itens.
2. Support Count: frequência de ocorrência de um item-set
3. Support (s): frequencia relativa das transações que contém o item-set X

$$Support(X) = \frac{frequency(X)}{N}$$

Para a regra  $A \Rightarrow B$ , Support é dado por:

$$Support(A \Rightarrow B) = \frac{frequency(A, B)}{N}$$

Qual o Support para *leite*  $\Rightarrow$  *fralda*?

4. Confidence (c): para a regra  $A \Rightarrow B$ , confidence mostra a porcentagem em que  $B$  é comprado junto com  $A$ .

$$Confidence(A \Rightarrow B) = \frac{frequency(A, B)}{frequency(A)} = \frac{P(A \cap B)}{P(A)}$$

O número de transações em  $A$  e  $B$  dividido pelo número total de transações que contém  $A$ .

$$Confidence(pão \Rightarrow leite) = \frac{3}{4} = 0,75 = 75\%$$

Agora encontre a medida Confidence para *leite*  $\Rightarrow$  *fralda*

Support e confidence medem o quanto interessante é a regra. É definido pelo suporte mínimo e pelos limites mínimos de confiança. Esses limites definidos pelo cliente ajudam a comparar a força da regra de acordo com a sua vontade ou a do cliente. Quanto mais próximo do limite, mais a regra é usada pelo cliente.

5. Frequent Itemsets: quando você aplica conjuntos de itens de associação cujo suporte é maior ou igual ao limite (threshold) mínimo de suporte (min\_sup). No exemplo acima, min\_sup = 3. Isso é definido na escolha do usuário.

6. Strong rules: se a regra  $A \Rightarrow B$  [Support, confidence] satisfaz min\_sup e min\_confidence, então esta é uma regra forte.
7. Lift: a elevação (lift) fornece a associação entre  $A$  e  $B$  na regra  $A \Rightarrow B$ . A associação mostra como um conjunto de itens  $A$  afeta o conjunto de itens  $B$ .

$$Lift(A \Rightarrow B) = \frac{Support(A \Rightarrow B)}{Support(A)Support(B)}$$

Por exemplo, para a regra  $pão \Rightarrow leite$ , lift é calculado como

$$\begin{aligned} support(pão) &= \frac{4}{5} = 0,8 \\ support(leite) &= \frac{4}{5} = 0,8 \\ lift(pão \Rightarrow leite) &= \frac{0,6}{0,8 \times 0,8} = 0,9 \end{aligned}$$

- Se a regra teve um lift de 1,  $A$  e  $B$  são independentes e nenhuma regra pode ser derivada deles.
- Se o lift for  $> 1$ ,  $A$  e  $B$  dependem um do outro, e o grau é dado pelo valor lift.
- Se o lift for  $< 1$ , a presença de  $A$  terá um efeito negativo em  $B$ .

### Mas qual o objetivo da Mineração de Regras de Associação?

Quando você aplica a Mineração de Regras de Associação em um determinado conjunto de transações, seu objetivo é encontrar todas as regras com:

- Suporte maior ou igual a min\_support
- Confiança maior ou igual a min\_confidence

## APRIORI Algorithm

O Association Rule Mining é vista como uma abordagem em duas etapas:

1. Geração frequente de conjuntos de itens: encontra-se todos os conjuntos de itens frequentes com suporte  $\geq$  contagem pré-determinada de min\_support
2. Geração de regras: lista-se todas as regras de associação dos conjuntos de itens frequentes. Calcula-se o Suporte e Confiança para todas as regras e remove-se regras que falham nos limites min\_support e min\_confidence.

A Geração frequente de itens é a etapa mais cara do ponto de vista computacional, pois requer uma verificação completa do banco de dados.

Entre as etapas acima, a geração de conjunto de itens frequentes é a mais cara em termos de cálculo.

Acima, vimos o exemplo de apenas 5 transações, mas, no mundo real, os dados de transações para varejo podem exceder GB e TBs de dados para os quais é necessário um algoritmo otimizado para remover conjuntos de itens que não ajudarão nas etapas posteriores. Para isso, o algoritmo APRIORI é usado.

**“Qualquer subconjunto de um itemset frequente também deve ser frequente. Em outras palavras, nenhum superset de um itemset pouco frequente deve ser gerado ou testado”**

O algoritmo é representado no Itemset Lattice, que é uma representação gráfica do princípio do algoritmo APRIORI. Ele consiste no nó do conjunto de k itens e na relação de subconjuntos desse conjunto de itens k.

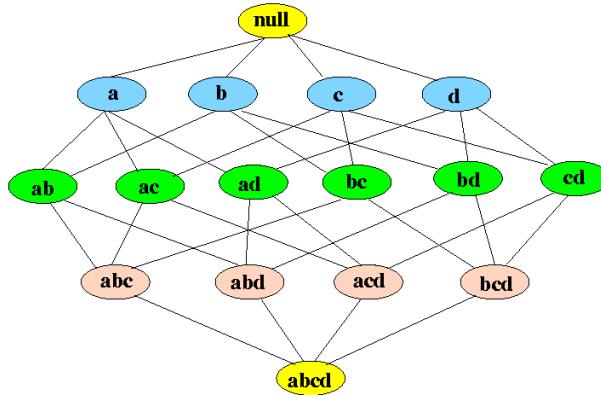


Figure 1: Itemset Lattice

Você pode ver na figura acima que na parte inferior estão todos os itens nos dados da transação e, em seguida, você começa a subir criando subconjuntos até o conjunto nulo. Para  $d$  número de itens, o tamanho da rede se tornará  $2^d$ . Isso mostra o quanto difícil será gerar o conjunto de itens frequentes, encontrando suporte para cada combinação. A figura a seguir mostra o quanto o APRIORI ajuda a reduzir o número de conjuntos a serem gerados:

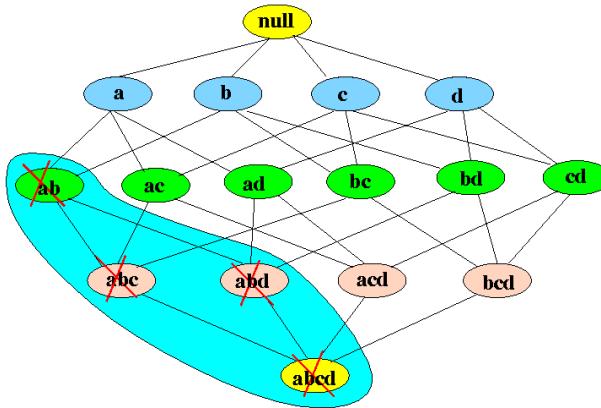


Figure 2: Itemset Lattice reduzido

Se o conjunto de itens  $\{a, b\}$  não for frequente, não precisamos levar em consideração todos os seus superconjuntos. Vamos entender isso por um exemplo. No exemplo a seguir, você verá por que o APRIORI é um algoritmo eficaz e também gera regras de associação fortes passo a passo. Pegue o seu caderno e caneta!

Transaction ID	Items		C1	Item set	Support count	L1	Item set	Support count
100	I1, I2		→	I1	3	→	I1	3
200	I2, I3, I4, I5			I2	4		I2	4
300	I2, I3			I3	3		I3	3
400	I1			I4	1			
500	I1, I2, I3			I5	1			

Transações da cesta de compras.  
Itens nomeados I1, I2, ...  
  
Definidos:  
min\_sup = 2  
min\_confidence = 50%

1. Inicia com todos os itens individuais candidatos e calcule as contagens de support  
Esta etapa chama-se **geração da lista de candidatos**

2. Remove os candidatos que falham na definição da contagem de min\_sup.  
Esta lista é chamada L1 e contém o frequent item set.  
**Aqui foi aplicado o princípio APRIORI**

Como você pode ver, comece criando a Lista de candidatos para o conjunto de 1 item que incluirá todos os itens presentes nos dados da transação, individualmente. Considerando os dados de transações de varejo do mundo real, você pode ver o quanto essa geração de candidatos é cara. Aqui, o APRIORI desempenha seu papel e ajuda a reduzir o número da lista de candidatos, e regras úteis são geradas no final. Nas etapas a seguir, você verá como chegamos ao final da geração do Conjunto de itens frequentes, que é a primeira etapa da mineração de regras de associação.

Item set	Support count	L2	Item set	Support count	C3	Item set	Support count
I1, I2	2	→	I1, I2	2	→	I1, I2, I3	1
I1, I3	1		I2, I3	3			
I2, I3	3						
I3 I1	1						

3. Gera a segunda lista de candidatos, cruzando L1 x L1 e anote a contagem de support

4. Remova os candidatos que falham em min\_sup.

5. Gera a terceira lista de candidatos, cruzando L2 x L2 e anote a contagem de support {I1, I2, I3} aparece em apenas uma lista juntos

6. L3 é nulo uma vez que a contagem de support para {I1, I2, I3} falha em min\_sup.  
Association rule mining está completa e não há uma lista de candidatos C4

Seu próximo passo será listar todos os conjuntos de itens frequentes. Você fará o último conjunto de itens frequentes não vazio, que neste exemplo é L2 = {I1, I2}, {I2, I3}. Em seguida, faça todos os subconjuntos não vazios dos conjuntos de itens presentes nessa lista de conjuntos de itens frequentes. Siga como mostrado na ilustração abaixo:

Item set	Support count	Nom-empty Subsets	Association Rules	Confidence
I1, I2	2	I2	{I2}=>I1	2/4=50% ✓
I2, I3	3	I1	{I1}=>I2	2/3=66% ✓
Nom-empty Subsets				✓ significa que todos limites foram satisfeitos min_confidence >=50% min_support count = 2
Association Rules	Confidence			
{I2}=>I3	3/4=75%	✓		
{I3}=>I2	3/3=100%	✓		

Dado min-confidence=50%. Confidence calculado como:  
 $c(A \Rightarrow B) = \text{support}(A, B) / \text{support}(A)$

**Confidence é o número de vezes A e B estão juntos em todas as transações contendo A**

Você pode ver acima, existem quatro regras fortes. Por exemplo, considere I2 => I3 ter confiança igual a 75% indica que 75% das pessoas que compraram I2 também compraram I3. Agora você aprendeu um

algoritmo APRIORI completo, que é um dos algoritmos mais usados na mineração de dados. Vamos ao código, phewww!

## Implementando MBA / Mineração de Regras de Associação usando R

Vamos usar um conjunto de dados do UCI Machine Learning Repository. O conjunto de dados é chamado Online-Retail, e você pode baixá-lo <http://archive.ics.uci.edu/ml/index.php>. O conjunto de dados contém dados de transação de 01/12/2010 a 09/12/2011 para um varejo on-line registrado não pertencente a lojas, com sede no Reino Unido. O motivo para usar este e não o conjunto de dados R é que você tem mais chances de receber dados de varejo neste formulário, nos quais terá que aplicar o pré-processamento de dados.

### Descrição do conjunto de dados

- Número de linhas: 541909
- Número de atributos: 08

Informações sobre atributos

- InvoiceNo: número da fatura. Nominal, um número integral de 6 dígitos atribuído exclusivamente a cada transação. Se este código começar com a letra ‘c’, indica um cancelamento.
- StockCode: código do produto (item). Nominal, um número integral de 5 dígitos atribuído exclusivamente a cada produto distinto.
- Descrição: nome do produto (item). Nominal.
- Quantidade: as quantidades de cada produto (item) por transação. Numérico.
- InvoiceDate: Data e hora da fatura. Numérico, o dia e a hora em que cada transação foi gerada. Exemplo do conjunto de dados: 1/12/2010 8:26
- Preço unitário: preço unitário. Numérico, preço do produto por unidade em libras esterlinas.
- Identificação do cliente: número do cliente. Nominal, um número integral de 5 dígitos atribuído exclusivamente a cada cliente.
- País: nome do país. Nominal, o nome do país em que cada cliente reside.

### Carregando bibliotecas

Primeiro, você carregará as bibliotecas necessárias.

```
#install and load package arules
if (!require(arules)) install.packages('arules')
#install and load arulesViz
if (!require(arulesViz)) install.packages('arulesViz')
#install and load tidyverse
if (!require(tidyverse)) install.packages('tidyverse')
#install and load readxl
if (!require(readxl)) install.packages('readxl')
#install and load knitr
if (!require(knitr)) install.packages('knitr')
#load ggplot2 as it comes in tidyverse
```

```

if (!require(ggplot2)) install.packages('ggplot2')
#install and load lubridate
if (!require(lubridate)) install.packages('lubridate')
#install and load plyr
if (!require(plyr)) install.packages('plyr')
if (!require(dplyr)) install.packages('dplyr')

```

## Data Pre-processing

Use a função `read_excel(path to file)` para ler a base de dados em Excel no R.

```

#read excel into R dataframe
retail <- read_excel('Online_Retail.xlsx')
#complete.cases(data) will return a logical vector indicating which rows have no missing
# values. Then use the vector to get only rows that are complete using retail[,].
retail <- retail[complete.cases(retail), ]
#mutate function is from dplyr package. It is used to edit or add new columns to
# dataframe. Here Description column is being converted to factor column. as.factor
# converts column to factor column. %>% is an operator with which you may pipe values
# to another function or expression
retail %>% mutate(Description = as.factor(Description))

```

```

## # A tibble: 406,829 x 8
##   InvoiceNo StockCode Description Quantity InvoiceDate      UnitPrice
##   <chr>     <chr>    <fct>       <dbl> <dttm>        <dbl>
## 1 536365   85123A  WHITE HANG~       6 2010-12-01 08:26:00  2.55
## 2 536365   71053   WHITE META~       6 2010-12-01 08:26:00  3.39
## 3 536365   84406B  CREAM CUPI~       8 2010-12-01 08:26:00  2.75
## 4 536365   84029G  KNITTED UN~       6 2010-12-01 08:26:00  3.39
## 5 536365   84029E  RED WOOLLY~       6 2010-12-01 08:26:00  3.39
## 6 536365   22752   SET 7 BABU~       2 2010-12-01 08:26:00  7.65
## 7 536365   21730   GLASS STAR~       6 2010-12-01 08:26:00  4.25
## 8 536366   22633   HAND WARME~       6 2010-12-01 08:28:00  1.85
## 9 536366   22632   HAND WARME~       6 2010-12-01 08:28:00  1.85
## 10 536367  84879   ASSORTED C~      32 2010-12-01 08:34:00  1.69
## # ... with 406,819 more rows, and 2 more variables: CustomerID <dbl>,
## #   Country <chr>

retail %>% mutate(Country = as.factor(Country))

```

```

## # A tibble: 406,829 x 8
##   InvoiceNo StockCode Description Quantity InvoiceDate      UnitPrice
##   <chr>     <chr>    <chr>       <dbl> <dttm>        <dbl>
## 1 536365   85123A  WHITE HANG~       6 2010-12-01 08:26:00  2.55
## 2 536365   71053   WHITE META~       6 2010-12-01 08:26:00  3.39
## 3 536365   84406B  CREAM CUPI~       8 2010-12-01 08:26:00  2.75
## 4 536365   84029G  KNITTED UN~       6 2010-12-01 08:26:00  3.39
## 5 536365   84029E  RED WOOLLY~       6 2010-12-01 08:26:00  3.39
## 6 536365   22752   SET 7 BABU~       2 2010-12-01 08:26:00  7.65
## 7 536365   21730   GLASS STAR~       6 2010-12-01 08:26:00  4.25
## 8 536366   22633   HAND WARME~       6 2010-12-01 08:28:00  1.85
## 9 536366   22632   HAND WARME~       6 2010-12-01 08:28:00  1.85

```

```

## 10 536367     84879      ASSORTED C~          32 2010-12-01 08:34:00      1.69
## # ... with 406,819 more rows, and 2 more variables: CustomerID <dbl>,
## #   Country <fct>

#Converts character data to date. Store InvoiceDate as date in new variable
retail$date <- as.Date(retail$InvoiceDate)
#Extract time from InvoiceDate and store in another variable
TransTime<- format(retail$InvoiceDate,"%H:%M:%S")
#Convert and edit InvoiceNo into numeric
InvoiceNo <- as.numeric(as.character(retail$InvoiceNo))

#Bind new columns TransTime and InvoiceNo into dataframe retail
cbind(retail,TransTime)

cbind(retail,InvoiceNo)

#get a glimpse of your data
glimpse(retail)

## Observations: 406,829
## Variables: 9
## $ InvoiceNo    <chr> "536365", "536365", "536365", "536365", "5363...
## $ StockCode    <chr> "85123A", "71053", "84406B", "84029G", "84029E", "22752...
## $ Description  <chr> "WHITE HANGING HEART T-LIGHT HOLDER", "WHITE METAL LANT...
## $ Quantity     <dbl> 6, 6, 8, 6, 6, 2, 6, 6, 32, 6, 6, 8, 6, 6, 3, 2, 3, ...
## $ InvoiceDate  <dttm> 2010-12-01 08:26:00, 2010-12-01 08:26:00, 2010-12-01 0...
## $ UnitPrice    <dbl> 2.55, 3.39, 2.75, 3.39, 3.39, 7.65, 4.25, 1.85, 1.85, 1...
## $ CustomerID   <dbl> 17850, 17850, 17850, 17850, 17850, 17850, 17850, 17850, ...
## $ Country      <chr> "United Kingdom", "United Kingdom", "United Kingdom", "...
## $ Date         <date> 2010-12-01, 2010-12-01, 2010-12-01, 2010-12-01, 2010-1...

```

Agora, a dataframe de varejo contém 10 atributos, com dois atributos adicionais: Data e Hora.

Antes de aplicar a mineração de regra de associação/MBA, precisamos converter o quadro de dados em dados de transação para que todos os itens comprados juntos em uma fatura estejam em uma linha. Você pode ver no `glimpse` que cada transação está na forma atômica, ou seja, todos os produtos pertencentes a uma fatura são atônicos, como nos bancos de dados relacionais. Esse formato também é chamado como formato *singles*.

O que você precisa fazer é agrupar dados no dataframe de varejo por CustomerID, CustomerID e Date ou também pode agrupar dados usando InvoiceNo e Date. Para executar esse agrupamento, aplicamos uma função e armazenamos a saída em outro dataframe. Isso pode ser feito por `ddply`.

As seguintes linhas de código combinam todos os produtos de um InvoiceNo e date e combinam todos os produtos desse InvoiceNo e date como uma linha, com cada item, separado por “,”.

```

transactionData <- ddply(retail,c("InvoiceNo","Date"),
                           function(df1) paste(df1$Description,
                           collapse = ","))
#The R function paste() concatenates vectors to character and separated results using
# collapse=[any optional character string ]. Here ',' is used
# head(transactionData)

```

Em seguida, como InvoiceNo e Date não serão úteis na mineração de regras, você pode defini-las como NULL.

```

#set column InvoiceNo of dataframe transactionData
transactionData$InvoiceNo <- NULL
#set column Date of dataframe transactionData
transactionData$date <- NULL
#Rename column to items
colnames(transactionData) <- c("items")
#Show Dataframe transactionData
#transactionData

```

Esse formato para dados de transação é chamado de formato de cesta. Em seguida, você deve armazenar esses dados da transação em um arquivo .csv (valores separados por vírgula). Para isso, **write.csv()**

```
write.csv(transactionData, "market_basket_transactions.csv", quote = FALSE, row.names = FALSE)
```

Em seguida, você deve carregar esses dados da transação em um objeto da classe de transação. Isso é feito usando a função **read.transactions** do pacote arules no R.

```
tr <- read.transactions('market_basket_transactions.csv', format = 'basket', sep=',')
```

Quando você executa as linhas de código acima, pode obter muito EOF dentro da string citada em sua saída, não se preocupe. Verifique então os dados de transação.

```
tr
```

```

## transactions in sparse format with
## 22191 transactions (rows) and
## 7876 items (columns)

summary(tr)

## transactions as itemMatrix in sparse format with
## 22191 rows (elements/itemsets/transactions) and
## 7876 columns (items) and a density of 0.001930725
##
## most frequent items:
## WHITE HANGING HEART T-LIGHT HOLDER           REGENCY CAKESTAND 3 TIER
##                                         1803                               1709
## JUMBO BAG RED RETROSPOT                      PARTY BUNTING
##                                         1460                               1285
## ASSORTED COLOUR BIRD ORNAMENT                (Other)
##                                         1250                               329938
##
## element (itemset/transaction) length distribution:
## sizes
##   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16
## 3598 1594 1141 908 861 758 696 676 663 593 624 537 516 531 551 522
##   17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32
## 464  441  483  419  395  315  306  272  238  253  229  213  222  215  170  159
##   33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48
## 138  142  134  109  111  90  113  94  93  87  88  65  63  67  63  60
##   49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64

```

```

##   59   49   64   40   41   49   43   36   29   39   30   27   28   17   25   25
##   65   66   67   68   69   70   71   72   73   74   75   76   77   78   79   80
##   20   27   24   22   15   20   19   13   16   16   11   15   12   7    9   14
##   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95   96
##   15   12   8    9   11   11   14   8    6    5    6    11   6    4    4    3
##   97   98   99   100  101  102  103  104  105  106  107  108  109  110  111  112
##   6    5    2    4    2    4    4    3    2    2    6    3    4    3    2    1
##  113  114  116  117  118  120  121  122  123  125  126  127  131  132  133  134
##   3    1    3    3    3    1    2    2    1    3    2    2    1    1    2    1
##  140  141  142  143  145  146  147  150  154  157  168  171  177  178  180  202
##   1    2    2    1    1    2    1    1    3    2    2    2    1    1    1    1
##  204  228  236  249  250  285  320  400  419
##   1    1    1    1    1    1    1    1    1
##
##      Min. 1st Qu. Median     Mean 3rd Qu. Max.
##      1.00   3.00  10.00  15.21  21.00 419.00
##
## includes extended item information - examples:
##           labels
## 1          1 HANGER
## 2      10 COLOUR SPACEBOY PEN
## 3 12 COLOURED PARTY BALLOONS

```

O **summary(tr)** é um comando muito útil que nos fornece informações sobre nosso objeto de transação. Vamos dar uma olhada no que a saída acima diz:

- Existem 22191 transações (linhas) e 7876 itens (colunas). Observe que 7876 é a descrição do produto envolvida no conjunto de dados e as transações 22191 são coleções desses itens.
- Densidade informa a porcentagem de células diferentes de zero em uma matriz esparsa. Você pode calcular como o número total de itens comprados dividido por um número possível de itens nessa matriz. Você pode calcular quantos itens foram comprados usando a densidade:  $22191 \times 7876 \times 0.001930725 = 337445$

Nota: Uma matriz esparsa ou vetor esparso é uma matriz na qual a maioria dos elementos é zero. Por outro lado, se a maioria dos elementos for diferente de zero, a matriz será considerada densa. O número de elementos com valor zero dividido pelo número total de elementos é chamado de esparsidade da matriz (que é igual a 1 menos a densidade da matriz).

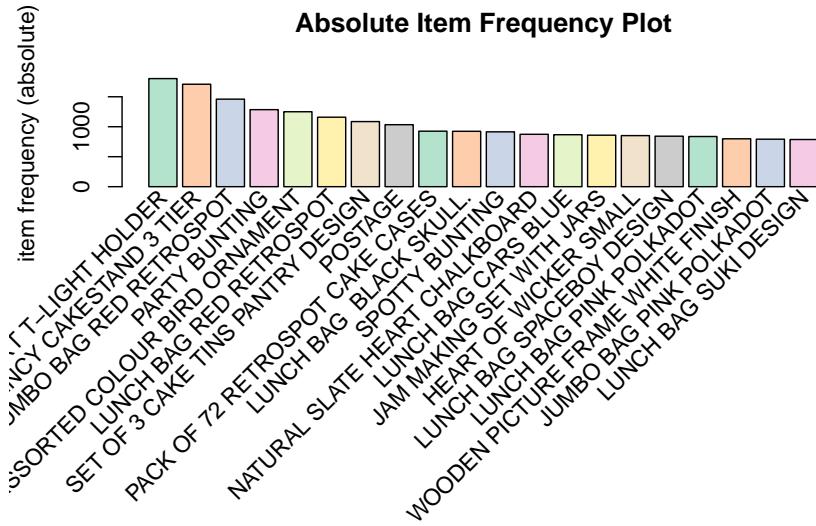
- summary também pode informar os itens mais frequentes.
- Distribuição do comprimento do elemento (conjunto de itens/transação): indica quantas transações existem para 1 conjunto de itens, 2 conjuntos de itens e assim por diante. A primeira linha indica um número de itens e a segunda linha indica o número de transações. Por exemplo, há apenas 3598 transações para um item, 1594 transações para 2 itens e 419 itens em uma transação que é a mais longa. Você pode gerar um itemFrequencyPlot para criar um item Gráfico de barras de frequência para visualizar a distribuição de objetos com base no **itemMatrix** (por exemplo, >transações ou itens em >itemsets e >rules), que é o nosso caso.

```

# Create an item frequency plot for the top 20 items
if (!require("RColorBrewer")) {
  # install color package of R
  install.packages("RColorBrewer")
}

```

```
#include library RColorBrewer
library(RColorBrewer)
}
itemFrequencyPlot(tr,topN=20,type="absolute",col=brewer.pal(8,'Pastel2'),
                 main="Absolute Item Frequency Plot")
```



No `itemFrequencyPlot(tr, topN = 20, tipo = "absoluto")`, o primeiro argumento é o objeto de transação a ser plotado que é tr. topN permite plotar N itens de frequência mais alta. type pode ser type = "absoluto" ou type = "relativo". Se absoluto, plotará as frequências numéricas de cada item independentemente. Se relativo, ele plotará quantas vezes esses itens apareceram em comparação com outros.

Esse gráfico mostra que 'T-LIGHT HEART WHITE HANGING HEART' e 'REGENCY CAKESTAND 3 TIER' têm mais vendas. Portanto, para aumentar a venda de 'SET OF 3 CAKE TINS PANTRY DESIGN', o revendedor pode colocá-lo perto de 'REGENCY CAKESTAND 3 TIER'.

O próximo passo é analisar as regras usando o algoritmo APRIORI. A função `apriori()` está no pacote arules.

```
# Min Support as 0.001, confidence as 0.8.
association.rules <- apriori(tr, parameter = list(supp=0.001, conf=0.8,maxlen=10))
```

```
## 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: 22
##
## set item appearances ...[0 item(s)] done [0.00s].
```

```

## set transactions ... [7876 item(s), 22191 transaction(s)] done [0.22s].
## sorting and recoding items ... [2324 item(s)] done [0.01s].
## creating transaction tree ... done [0.02s].
## checking subsets of size 1 2 3 4 5 6 7 8 9 10 done [0.44s].
## writing ... [49122 rule(s)] done [0.05s].
## creating S4 object ... done [0.04s].

```

A função **apriori** tomará tr como o objeto de transação no qual a mineração será aplicada. O parâmetro permitirá que você defina min\_sup e min\_confidence. Os valores padrão para o parâmetro são suporte mínimo de 0,1, confiança mínima de 0,8 e máximo de 10 itens (maxlen).

**summary(association.rules)** mostra o seguinte:

- Especificação de parâmetro: min\_sup = 0.001 e min\_confidence = 0.8 valores com 10 itens como o máximo de itens em uma regra.
- Número total de regras: o conjunto de 49122 regras
- Distribuição do tamanho da regra: Um tamanho de 5 itens tem mais regras: 16424 e o tamanho de 2 itens tem o menor número de regras: 105
- Resumo das medidas de qualidade: valores mínimo e máximo de Suporte, Confiança e Elevação.
- Informações usadas para criar regras: os dados, suporte e confiança que fornecemos ao algoritmo.

Como existem 49122 regras, vamos imprimir apenas as 10 principais:

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

##	lhs	rhs	support	confidence	lift	co
## [1]	{WOBBLY CHICKEN}	=> {DECORATION}	0.001261773	1.0000000	443.8200	
## [2]	{WOBBLY CHICKEN}	=> {METAL}	0.001261773	1.0000000	443.8200	
## [3]	{DECOUPAGE}	=> {GREETING CARD}	0.001036456	1.0000000	389.3158	
## [4]	{BILLBOARD FONTS DESIGN}	=> {WRAP}	0.001306836	1.0000000	715.8387	
## [5]	{WRAP}	=> {BILLBOARD FONTS DESIGN}	0.001306836	0.9354839	715.8387	
## [6]	{ENAMEL PINK TEA CONTAINER}	=> {ENAMEL PINK COFFEE CONTAINER}	0.001396963	0.8157895	385.1741	
## [7]	{WOBBLY RABBIT}	=> {DECORATION}	0.001532153	1.0000000	443.8200	
## [8]	{WOBBLY RABBIT}	=> {METAL}	0.001532153	1.0000000	443.8200	
## [9]	{ART LIGHTS}	=> {FUNK MONKEY}	0.001712406	1.0000000	583.9737	
## [10]	{FUNK MONKEY}	=> {ART LIGHTS}	0.001712406	1.0000000	583.9737	

Usando a saída acima, você pode fazer análises como:

- 100% dos clientes que compraram ‘WOBBLY CHICKEN’ também compraram ‘METAL’.
- 100% dos clientes que compraram ‘BLACK TEA’ também compraram ‘AÇÚCAR’ JARS’.

### Limitar o número e tamanho das regras

Como você pode limitar o tamanho e o número de regras geradas? Você pode fazer isso definindo parâmetros em apriori. Você define esses parâmetros para ajustar o número de regras que receberá. Se você deseja regras mais fortes, pode aumentar o valor de conf e, para regras mais estendidas, dar maior valor ao maxlen.

```

shorter.association.rules <- apriori(tr, parameter = list(supp=0.001, conf=0.8,maxlen=3))

## 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
##           3  rules FALSE
##
## Algorithmic control:
##   filter tree heap memopt load sort verbose
##           0.1 TRUE TRUE FALSE TRUE     2    TRUE
##
## Absolute minimum support count: 22
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[7876 item(s), 22191 transaction(s)] done [0.18s].
## sorting and recoding items ... [2324 item(s)] done [0.01s].
## creating transaction tree ... done [0.02s].
## checking subsets of size 1 2 3 done [0.21s].
## writing ... [2216 rule(s)] done [0.04s].
## creating S4 object ... done [0.02s].

```

Você pode remover regras redundantes que são subsets de regras maiores usando o seguinte comando:

```

subset.rules <- which(colSums(is.subset(association.rules, association.rules)) > 1)
# get subset rules in vector
length(subset.rules) #> 3913

```

```
## [1] 44014
```

```

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

```

- `which()` retorna a posição dos elementos no vetor cujo valor é TRUE.
- `colSums()` forma uma soma de linhas e colunas para quadros de dados e matrizes numéricas.
- `is.subset()` Determina se os elementos de um vetor contêm todos os elementos de outro vetor

### Buscando regras de determinados elementos

Às vezes, você deseja trabalhar em um produto específico. Se você deseja descobrir o que causa influência na compra do item X, pode usar a opção de `appearance` no comando `apriori`. `appearance` nos dá opções para definir LHS (IF part) e RHS (then part) da regra. Por exemplo, para descobrir o que os clientes compram antes de comprar ‘METAL’, execute a seguinte linha de código:

```

metal.association.rules <- apriori(tr, parameter = list(supp=0.001, conf=0.8),
                                    appearance = list(default="lhs",rhs="METAL"))

```

```

## 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: 22
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[7876 item(s), 22191 transaction(s)] done [0.21s].
## sorting and recoding items ... [2324 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 [0.42s].
## writing ... [5 rule(s)] done [0.07s].
## creating S4 object ... done [0.02s].

```

```

# Here lhs=METAL because you want to find out the probability of that in how many
# customers buy METAL along with other items
inspect(head(metal.association.rules))

```

	lhs	rhs	support	confidence	lift	count
## [1]	{WOBBLY CHICKEN}	=> {METAL}	0.001261773	1	443.82	28
## [2]	{WOBBLY RABBIT}	=> {METAL}	0.001532153	1	443.82	34
## [3]	{DECORATION}	=> {METAL}	0.002253166	1	443.82	50
## [4]	{DECORATION,WOBBLY CHICKEN}	=> {METAL}	0.001261773	1	443.82	28
## [5]	{DECORATION,WOBBLY RABBIT}	=> {METAL}	0.001532153	1	443.82	34

Da mesma forma, para encontrar a resposta para a pergunta Os clientes que compraram METAL também compraram .... você manterá o METAL em lhs:

```

metal.association.rules <- apriori(tr, parameter = list(supp=0.001, conf=0.8),
                                     appearance = list(lhs="METAL",default="rhs"))

```

```

## 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: 22
##

```

```

## set item appearances ... [1 item(s)] done [0.00s].
## set transactions ... [7876 item(s), 22191 transaction(s)] done [0.18s].
## sorting and recoding items ... [2324 item(s)] done [0.01s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 done [0.01s].
## writing ... [1 rule(s)] done [0.00s].
## creating S4 object ... done [0.02s].
```

```
inspect(head(metal.association.rules))
```

```

##      lhs          rhs       support     confidence lift   count
## [1] {METAL} => {DECORATION} 0.002253166 1           443.82 50
```

## Visualizando Regras de Associação

Como haverá centenas ou milhares de regras geradas com base em dados, você precisará de duas maneiras de apresentar suas descobertas. O **ItemFrequencyPlot** já foi discutido acima, o que também é uma ótima maneira de obter os itens mais vendidos.

As seguintes visualizações serão discutidas:

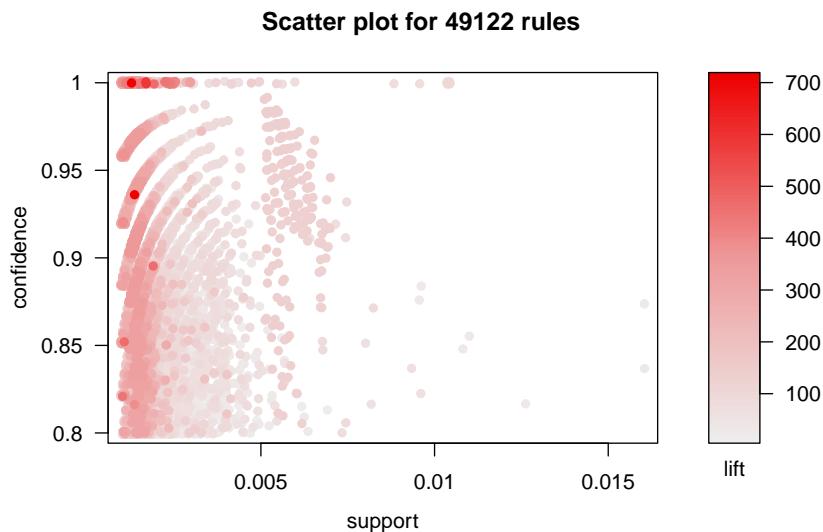
- Gráfico de dispersão
- Gráfico de dispersão interativo
- Representação de regra individual

### Gráfico de dispersão

Uma visualização direta das regras de associação é usar um gráfico de dispersão usando **plot()** do pacote arulesViz. Ele usa Suporte e Confiança nos eixos. Além disso, a terceira medida de elevação é usada por padrão para colorir (níveis de cinza) dos pontos.

```

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

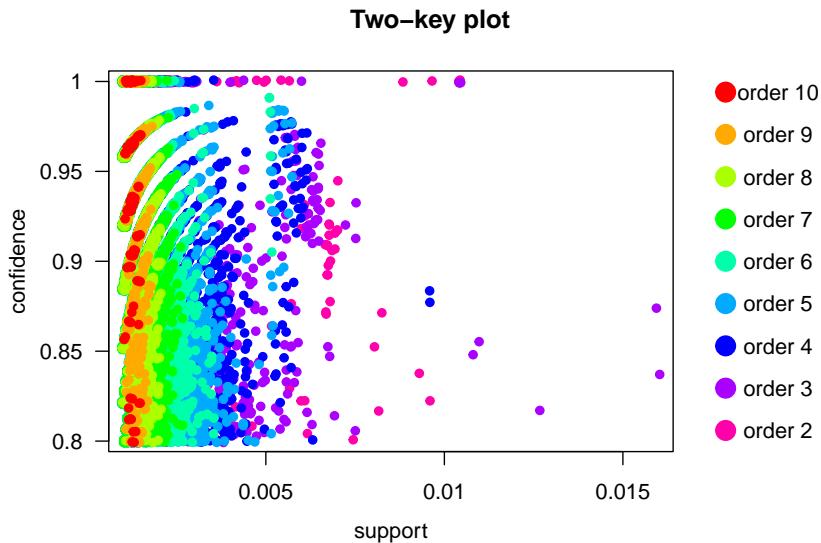


O gráfico acima mostra que regras com alta elevação têm baixo suporte. Você pode usar as seguintes opções para a plotagem:

```
plot(rulesObject, measure, shading, method)
```

- rulesObject: o objeto de regras a ser plotado
- measure: medidas para o interesse das regras. Pode ser Suporte, Confiança, lift ou combinação destes dependendo do valor do método.
- shading: Medida usada para colorir pontos (suporte, confiança, lift). O padrão é elevador.
- method: método de visualização a ser usado (scatterplot, two-key plot, matrix3D).

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



O two-key plot usa suporte e confiança nos eixos x e y, respectivamente. Ele usa ordem para colorir. Ordem é o número de itens na regra.

### Gráfico interativo de dispersão

Um plot interativo incrível pode ser usado para apresentar suas regras que usam arulesViz e plotly. Você pode passar o mouse sobre cada regra e visualizar todas as medidas de qualidade (suporte, confiança e lift).

```
plotly_arules(subRules)
```

### Visualizações baseadas em gráficos

Técnicas baseadas em gráficos visualizam regras de associação usando vértices e arestas, onde os vértices são rotulados com nomes de itens e conjuntos ou regras de itens são representados como um segundo conjunto de vértices. Os itens são conectados aos conjuntos / regras de itens usando setas direcionadas. As setas apontando dos itens para os vértices da regra indicam itens do LHS e uma seta de uma regra para um item indica o RHS. O tamanho e a cor dos vértices geralmente representam medidas de interesse. Os gráficos são uma ótima maneira de visualizar regras, mas tendem a ficar congestionados à medida que o número de regras aumenta. Portanto, é melhor visualizar menos número de regras com visualizações baseadas em gráficos. Vamos selecionar 10 regras das sub-regras com a maior confiança e depois plotar um gráfico interativo.

```
top10subRules <- head(subRules, n = 10, by = "confidence")
plot(top10subRules, method = "graph", engine = "htmlwidget")
```

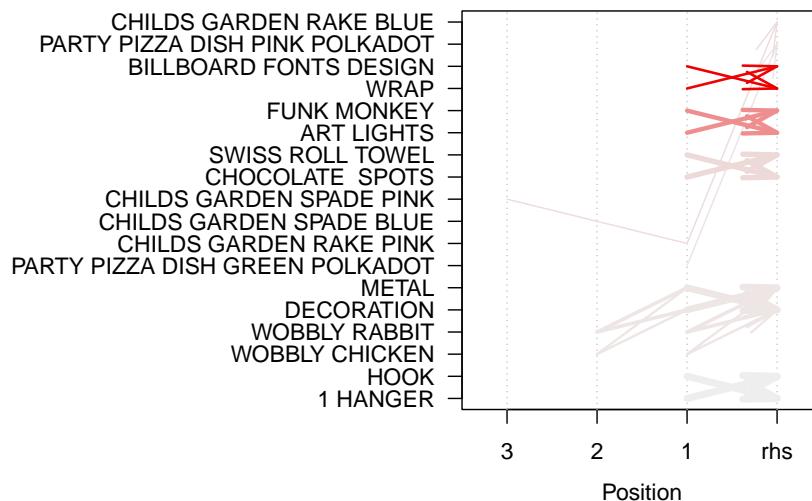
Nota: Você pode tornar todos os seus gráficos interativos usando o parâmetro engine = htmlwidget no gráfico.

### Representação de regras individuais

Essa representação também é chamada como gráfico de coordenadas paralelas. É útil visualizar quais produtos e quais itens causam que tipo de vendas. Como mencionado acima, o RHS é o consequente ou o item que propomos que o cliente comprará; as posições estão no LHS, onde 2 é a adição mais recente à nossa cesta e 1 é o item que tínhamos anteriormente.

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

**Parallel coordinates plot for 20 rules**



Olhe para a seta mais acima. Isso mostra que, quando tenho 'CHILDS GARDEN SPADE PINK' e 'CHILDS GARDEN RAKE PINK' em meu carrinho de compras, provavelmente comprarei 'CHILDS GARDEN RAKE BLUE' junto com elas também.

### Conclusões

Parabéns! Executamos o APRIORI, um dos algoritmos mais frequentemente usados na mineração de dados. Aprendemos o necessário sobre a Association Rule Mining, seus aplicativos e seus aplicativos no varejo, chamados de Market Basket Analysis. Agora você também é capaz de implementar a Análise de cesta de mercado em R e apresentar suas regras de associação com ótimos gráficos!