# Fundamentos de la Ciencia de Datos Práctica 2

Fernández Díaz, Daniel.
Cano Díaz, Francisco.
Fernández Hernández, Alberto.

22 de octubre del 2019

## 1. Apartado 1

El primer apartado consistirá en realizar un análisis de **asociación** de Datos con `R`, utilizando la muestra vista en teoría:

*{Pan, Agua, Leche, Naranjas}, {Pan, Agua, Café, Leche},*
*{Pan, Agua, Leche}, {Pan, Café, Leche}, {Pan, Agua}, {Leche}*

Para resolverlo, emplearemos el algoritmo ***a priori***, ubicado en el paquete ***arules***. Por defecto, este paquete no se encuentra instalado en el entorno de `R`, por lo que debemos descargarlo e instalarlo. Para ello, debemos seguir los siguientes pasos:

1. En primer lugar, nos dirigimos a la página del proyecto ***CRAN***, donde seleccionaremos la opción ***packages***.
2. A continuación, seleccionamos ***Table of available packages, sorted by name*** con la que podremos ver todos los paquetes disponibles ordenados por nombre.
3. Seleccionamos el paquete ***arules***.
4. Una vez situados en la página del paquete, debemos destacar varios campos:
   - **Archivos binarios**: para la instalación de la librería.
   - **Manual de referencia**: con toda la información sobre las funciones del paquete.
   - **URL**: repositorio en ***GitHub*** del paquete.
5. Descargamos el archivo *.zip* (preferiblemente la opción ***release***).
6. Una vez descargado, creamos un directorio en *C:*, llamado *tmp*, donde añadiremos el archivo *.zip*.

**Finalmente, ejecutamos el siguiente comando:**

```
> #Instalamos el paquete
> install.packages("C:\\tmp\\arules_1.6-4.zip", repos=NULL)

> #Finalmente, lo importamos con library
> library(arules)
```

Una vez importado el paquete, debemos introducir las muestras en `R`. Para ello convertiremos las muestras en una matriz de unos y ceros:

| | Pan | Agua | Cafe | Leche | Naranjas |
|---|---|---|---|---|---|
| suceso 1 | 1 | 1 | 0 | 1 | 1 |
| suceso 2 | 1 | 1 | 1 | 1 | 0 |
| suceso 3 | 1 | 1 | 0 | 1 | 0 |
| suceso 4 | 1 | 0 | 1 | 1 | 0 |
| suceso 5 | 1 | 1 | 0 | 0 | 0 |
| suceso 6 | 0 | 0 | 0 | 1 | 0 |

Para crear una matriz en `R` utilizaremos la función ***Matrix***, la cual ya disponemos en nuestro ***workspace*** al importar la librería ***arules***:

```
> muestra <- Matrix(c(1,1,0,1,1,1,1,1,1,0,1,1,0,1,0,1,0,1,1,0,1,1,0,0,0,0,0,0,1,0),6,5,
+ byrow=T,dimnames=list(c("suceso1","suceso2","suceso3","suceso4","suceso5","suceso6"),
+ c("Pan","Agua","Cafe","Leche","Naranjas")),sparse=T)
> muestra
```

```
6 x 5 sparse Matrix of class "dgCMatrix"
        Pan Agua Cafe Leche Naranjas
suceso1   1    1    .     1        1
suceso2   1    1    1     1        .
suceso3   1    1    .     1        .
suceso4   1    .    1     1        .
suceso5   1    1    .     .        .
suceso6   .    .    .     1        .
```

Una vez creada la matriz, vamos a convertirla en una matriz de tipo *binaria*, únicamente con entradas **TRUE**,**FALSE**. Para ello, utilizaremos la función ***nsparseMatrix***, para convertirla en una matriz de tipo binaria:

- | =>valores a 1
- . =>valores a 0

```
> muestrangCMatrix <- as(muestra, "nsparseMatrix")
> muestrangCMatrix
```

```
6 x 5 sparse Matrix of class "ngCMatrix"
        Pan Agua Cafe Leche Naranjas
suceso1   |    |    .     |        |
suceso2   |    |    |     |        .
suceso3   |    |    .     |        .
suceso4   |    .    |     |        .
suceso5   |    |    .     .        .
suceso6   .    .    .     |        .
```

A continuación, creamos la matriz traspuesta de ***muestrangCMatrix***, empleando la función ***t***:

```
> traspuestangCMatrix <- t(muestrangCMatrix)
> traspuestangCMatrix
```

```
5 x 6 sparse Matrix of class "ngCMatrix"
         suceso1 suceso2 suceso3 suceso4 suceso5 suceso6
Pan            |       |       |       |       .
Agua           |       |       .       |       .
Cafe           .       |       .       |       .
Leche          |       |       |       .       |
Naranjas       |       .       .       .       .
```

Una vez obtenida traspuesta, debemos transformar la matriz en un tipo de datos específico para el paquete ***arules***: **transacciones**. Se trata de un tipo de dato específico para ***Data Mining***:

```
> transacciones  <- as(traspuestangCMatrix, "transactions")
> transacciones
```

```
transactions in sparse format with
 6 transactions (rows) and
 5 items (columns)
```

Analizando el resultado obtenido, podemos observar que disponemos de un total de **6 muestras** y **5 sucesos individuales**. Si utilizamos la función ***inspect***, podemos analizar en detalle cada una de las muestras:

```
> inspect(transacciones)
```

```
    items                      itemsetID
[1] {Pan,Agua,Leche,Naranjas} suceso1
[2] {Pan,Agua,Cafe,Leche}     suceso2
[3] {Pan,Agua,Leche}          suceso3
[4] {Pan,Cafe,Leche}          suceso4
[5] {Pan,Agua}                suceso5
[6] {Leche}                   suceso6
```

Por otro lado, la función ***summary*** nos proporciona un breve resumen de las transacciones:

```
> summary(transacciones)
```

```
transactions as itemMatrix in sparse format with
 6 rows (elements/itemsets/transactions) and
 5 columns (items) and a density of 0.5666667

most frequent items:
     Pan    Leche     Agua     Cafe Naranjas  (Other)
       5        5        4        2        1        0

element (itemset/transaction) length distribution:
sizes
1 2 3 4
1 1 2 2

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.250   3.000   2.833   3.750   4.000

includes extended item information - examples:
  labels
1    Pan
2   Agua
3   Cafe

includes extended transaction information - examples:
  itemsetID
1   suceso1
2   suceso2
3   suceso3
```

Finalmente, ejecutamos el algoritmo ***apriori***, indicando como parámetros:

- Las transacciones a procesar.
- El valor de soporte (en nuestro caso, será de un 50 %)
- El valor de confianza (en nuestro caso, de un 80 %)

```
> asociaciones <- apriori(transacciones, parameter=list(support=0.5, confidence=0.8))

Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.8    0.1    1 none FALSE            TRUE       5     0.5      1     10  rules FALSE

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

Absolute minimum support count: 3

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[5 item(s), 6 transaction(s)] done [0.00s].
sorting and recoding items ... [3 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [7 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].

> inspect(asociaciones)

    lhs             rhs     support   confidence lift count
[1] {}           => {Leche} 0.8333333 0.8333333  1.00 5
[2] {}           => {Pan}   0.8333333 0.8333333  1.00 5
[3] {Agua}       => {Pan}   0.6666667 1.0000000  1.20 4
[4] {Pan}        => {Agua}  0.6666667 0.8000000  1.20 4
[5] {Leche}      => {Pan}   0.6666667 0.8000000  0.96 4
[6] {Pan}        => {Leche} 0.6666667 0.8000000  0.96 4
[7] {Agua,Leche} => {Pan}   0.5000000 1.0000000  1.20 3
```

Como podemos observar, el algoritmo ***apriori*** muestra aquellas asociaciones cuyos valores tanto de **soporte** como de **confianza** superan el umbral establecido al comienzo:

$$\{Agua,\ Pan\},\ \{Pan,\ Agua\}$$
$$\{Leche,\ Pan\},\ \{Pan,\ Leche\},\ \{Agua,\ Leche,\ Pan\}$$

Por otro lado, el campo ***lift*** nos indica **la proporción entre la confianza de una asociación y el soporte del segundo suceso**: ¿Para qué se calcula este campo? Tenemos la siguiente asociación:

$$\{Leche\} \rightarrow \{Pan\}$$

Supongamos que la confianza de la asociación anterior es del 70 %, es decir, sabemos que el **70 % de las personas que compran Leche compran también Pan**. Sin embargo, el soporte del segundo elemento es también del 70 %, esto es, **el 70 % de las personas que van a comprar a un supermercado compran Pan**. Por tanto, el hecho de que el 70 % de las ocasiones se compre Pan cuando se compra Leche **no me está aportando información relevante cuando sabemos que en el 70 % de las ocasiones se compra Pan.** En conclusión: en ocasiones, valores muy altos de confianza pueden deberse a que el suceso de la derecha tenga un valor de soporte elevado:

- Para valores de ***lift*** mayores que 1 significa que **la probabilidad del producto de la derecha de la asociación aumenta conforme se compra el producto de la izquierda**. La confianza nos proporciona información relevante.
- Para valores de ***lift*** iguales a 1 implica que la probabilidad del producto de la derecha de la asociación no se ve afectada una vez comprados los productos de la izquierda de la asociación. Como consecuencia, **la confianza entre ambos sucesos no nos está proporcionando información relevante**.
- Para valores de ***lift*** menores que 1 significa que **la probabilidad del producto de la derecha de la asociación disminuye conforme se compra el producto de la izquierda**. Al igual que en el primer caso, la confianza nos proporciona información relevante.

Analizando de nuevo los resultados obtenidos tras ejecutar ***inspect***:

```
    lhs              rhs     support   confidence lift count
[1] {}           => {Leche} 0.8333333 0.8333333  1.00 5
[2] {}           => {Pan}   0.8333333 0.8333333  1.00 5
[3] {Agua}       => {Pan}   0.6666667 1.0000000  1.20 4
[4] {Pan}        => {Agua}  0.6666667 0.8000000  1.20 4
[5] {Leche}      => {Pan}   0.6666667 0.8000000  0.96 4
[6] {Pan}        => {Leche} 0.6666667 0.8000000  0.96 4
[7] {Agua,Leche} => {Pan}   0.5000000 1.0000000  1.20 3
```

Podemos observar que el valor de ***lift*** para la asociación

$$\{Leche\} \rightarrow \{Pan\}$$

es de 0.96, es decir:

$$lift = \frac{c(A_1 \rightarrow A_2)}{s(A_2)} = \frac{0{,}8}{0{,}8333} = 0{,}96$$

[1]

Al ser menor que 1, **la probabilidad del producto Pan de la asociación disminuye conforme se compra Leche**, en relación al suceso individual Pan.

En cuanto a los **resultados del algoritmo**, podemos observar que son los mismos que los obtenidos en teoría y que efectivamente si comprobasemos el soporte y la confianza con la fórmula de cada una de las asociaciones nos daría el mismo porcentaje que se muestra en la tabla anterior. Además, con **estas asociaciones podemos concluir** que cuando se compra Agua se compra Pan con una confianza del 100 % o cuando se compra Agua y Leche también se compra Pan con una confianza del 100 %.

[1] el soporte individual para el pan es de 5/6

# 2. Apartado 2

## 2.1. Apartado 2.1

Realizaremos un análisis de **asociación** de Datos con `R`, extrayendo los datos de un **fichero *.txt***. Los datos que tenemos son los siguientes:

$$\{X,\ C,\ N,\ B\},\ \{X,\ T,\ B,\ C\},\ \{N,\ C,\ X\},\ \{N,\ T,\ X,\ B\}$$
$$\{X,\ C,\ B\},\ \{N\},\ \{X,\ B,\ C\},\ \{T,\ A\}$$

Donde X: Faros de Xenon, A: Alarma, T: Techo Solar, N: Navegador, B: Bluetooth y C: Control de Velocidad. Estos son cada uno de los extras que se pueden incluir en cada coche.
Lo primero que debemos hacer es leer estos datos desde un fichero ***.txt***. Para ello lo que haremos será convertir estas muestras en una matriz binaria. Para ello tenemos dos opciones:

### 2.1.1. Lectura de la matriz binaria desde el fichero *.txt*

Leemos la siguiente matriz binaria del fichero *.txt*:

| | X | A | T | N | B | C |
|---|---|---|---|---|---|---|
| suceso 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| suceso 2 | 1 | 0 | 1 | 0 | 1 | 1 |
| suceso 3 | 1 | 0 | 0 | 1 | 0 | 1 |
| suceso 4 | 1 | 0 | 1 | 1 | 1 | 0 |
| suceso 5 | 1 | 0 | 0 | 0 | 1 | 1 |
| suceso 6 | 0 | 0 | 0 | 1 | 0 | 0 |
| suceso 7 | 1 | 0 | 0 | 0 | 1 | 1 |
| suceso 8 | 0 | 1 | 1 | 0 | 0 | 0 |

Para leer el fichero *.txt* utilizaremos el comando ***read.table***:

```
> muestra <- read.table("binariaCoches.txt")
> muestra

  X A T N B C
1 1 0 0 1 1 1
2 1 0 1 0 1 1
3 1 0 0 1 0 1
4 1 0 1 1 1 0
5 1 0 0 0 1 1
6 0 0 0 1 0 0
7 1 0 0 0 1 1
8 0 1 1 0 0 0
```

Una vez leido los datos procederemos a aplicar el **Algoritmo Apriori** utilizando unas funciones que nos prepararán las muestras para que puedan ser leidas correctamente por el comando ***apriori***:
Primero debemos crear una función que nos convierta el *dataframe* leído en una matriz utilizando el comando ***Matrix***:

```
> matriz.binaria.dgC <- function(dataframe){
+         aux=dataframe[,1]
+         for(i in 2:length(colnames(dataframe))){
+                 aux=c(aux,dataframe[,i])
+         }
+         Matrix(aux,length(rownames(dataframe)),length(colnames(dataframe)),
+                byrow=F,dimnames=dimnames(dataframe),sparse=T)
+ }
```

Después debemos convertir esa matriz en una matriz binaria utilizando la función ***nsparseMatrix***:

```
> matriz.binaria.ngC <- function(dataframe){
+         as(matriz.binaria.dgC(dataframe),"nsparseMatrix")
+ }
```

Una vez tenemos la matriz binaria, debemos hacer su traspuesta utilizando la función ***t***:

```
> matriz.binaria.ngc.traspuesta <- function(ngCMatrix){
+         t(ngCMatrix)
+ }
```

Ahora convertimos esa matriz traspuesta en transacciones, utilizando la función ***transactions***, que es el tipo de datos necesario para la función *apriori*:

```
> matriz.transaccion <- function(ngCMatrixTras){
+         as(ngCMatrixTras, "transactions")
+ }
```

Una vez construidas todas las funciones anteriores para modelar los datos antes de pasarselos al **Algoritmo Apriori**, ejecutaremos el algoritmo a través de la siguiente función:

```
> algoritmo.apriori <- function(datos,soporte,confianza){
+         mb <- matriz.binaria.ngC(datos)
+         mbt <- matriz.binaria.ngc.traspuesta(mb)
+     transacciones <- matriz.transaccion(mbt)
+         #Al comando apriori le pasamos las transacciones, el soporte y la confianza
+         asociaciones <- apriori(transacciones,parameter=list(support=soporte,confidence=confianza))
+         inspect(asociaciones)
+ }
> #En nuestro caso estableceremos un soporte del 40% y una confianza del 90%
> algoritmo.apriori(muestra,0.4,0.9)

Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.9    0.1    1 none FALSE            TRUE       5     0.4      1     10  rules FALSE

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

Absolute minimum support count: 3
```

```
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[6 item(s), 8 transaction(s)] done [0.00s].
sorting and recoding items ... [4 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [3 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
    lhs      rhs support confidence lift     count
[1] {C}   => {X} 0.625   1          1.333333 5
[2] {B}   => {X} 0.625   1          1.333333 5
[3] {B,C} => {X} 0.500   1          1.333333 4
```

### 2.1.2. Lectura de las muestras desde el fichero *.txt*

Para leer las muestras directamente de un fichero *.txt* es necesario:

- No usar llave.
- Cada suceso debe ir en una línea del fichero.
- Cada suceso elemental debe estar separado por un espacio en blanco u otro tipo de separador.
- Al final del fichero debe haber un *enter*.

De esta forma el fichero nos quedaría de la siguiente forma:

X C N B
X T B C
N C X
N T X B
X C B
N
X B C
T A

Una vez que tenemos el fichero, nos disponemos a leerlo de la siguiente forma:

```
> # Algoritmo Apriori a partir de una muestra
> muestra.apriori <- function(ruta, sep=" ", soporte=0.5, confianza=0.8, verbose=F){
+         cat("___ ALGORITMO A PRIORI _____________________________________________\n\n")
+         datos <- muestra.leer(ruta,sep)
+         matriz <- muestra.procesar(datos)
+         if(verbose){
+                 cat("===MATRIZ NGC DE LA MUESTRA=============================================\n")
+                 print(matriz)
+                 cat("=========================================================================\n\n")
+         }
+     transacciones <- matriz.transaccion(matriz.binaria.ngc.traspuesta(matriz))
+         if(verbose){
+                 cat("===TRANSACCIONES=========================================================\n")
+                 inspect(transacciones)
+                 cat("=========================================================================\n\n")
+         }
+         asociaciones <- apriori(transacciones,parameter=list(support=soporte,confidence=confianza))
+         cat("\n===ASOCIACIONES=========================================================\n")
+         inspect(asociaciones)
+         cat("=========================================================================\n\n")
+ }
> # Leer archivo muestra:
> # Mediante la ruta y la separacion entre sucesos elementales
> # creamos un flujo de datos (conexion) con el fichero y volcamos
> # la información en una variable llamada datos y luego con la funcion
> # strsplit separamos cada linea del fichero en elementos individuales
> # a partir del separador.
> muestra.leer <- function(ruta,sep=" "){
+         con <- file(ruta, "r", blocking = FALSE)
+         datos <- readLines(con)
+         close(con)
+         strsplit(datos,sep)
+ }
> # Procesar la muestra:
> # 1. Identifica todos los sucesos elementales diferentes.
> # 2. Obtenemos los ceros y unos que representan cada suceso.
> # 3. Creamos la matriz a partir de los datos binarios y
> #    los nombres de los sucesos elementales.
> muestra.procesar <- function(lista){
+         lista.elem <- muestra.obtenerElem(lista)
+         datos.bin <- muestra.obtenerBin(lista,lista.elem)
+         matriz <- muestra.matriz.ngC(datos.bin,lista.elem,length(lista),length(lista.elem))
+ }
```

```
> # Obtener lista de sucesos elementales:
> # A partir del txt original construye una lista
> # formada por los sucesos elementales de la muestra.
> muestra.obtenerElem <- function(lista.original){
+         lista = NULL
+         for(i in 1:length(lista.original)){
+                 for(j in 1:length(lista.original[[i]])){
+                         if(!(is.element(lista.original[[i]][j],lista))){
+                                 lista <- c(lista,lista.original[[i]][j])
+                         }
+                 }
+         }
+         lista
+ }
> # Obtener lista binaria de apariciones en la muestra:
> # A partir de los sucesos elementales revisamos para
> # cada linea de la estructura datos si dicho suceso
> # elemental esta presente en ese suceso de la muestra.
> # Si esta es un 1 y si no, un 0.
> muestra.obtenerBin <- function(datos,lista.elem){
+         datos.bin = NULL
+         for(i in 1:length(datos)){
+                 for(j in 1:length(lista.elem)){
+                         if(is.element(lista.elem[j],datos[[i]])){
+                                 datos.bin <- c(datos.bin,1)
+                         } else{
+                                 datos.bin <- c(datos.bin,0)
+                         }
+                 }
+         }
+         datos.bin
+ }
> # Crear la matriz ngc
> muestra.matriz.ngC <- function(datos.bin, lista.elem, f, c){
+         aux <- Matrix(datos.bin,f,c,byrow=T,dimnames=list(1:f,lista.elem),sparse=T)
+         as(aux,"nsparseMatrix")
+ }
```

Una vez definidas todas las funciones ejecutamos la función principal:

```
> muestra.apriori("muestraCoche.txt",soporte=0.4,confianza=0.9,verbose=T)

___ ALGORITMO A PRIORI _____________________________________________

===MATRIZ NGC DE LA MUESTRA=================================================
8 x 6 sparse Matrix of class "ngCMatrix"
  X C N B T A
1 | | | | . .
2 | | . | | .
3 | | | . . .
4 | . | | | .
5 | | . | . .
6 . . | . . .
7 | | . | . .
8 . . . . | |
============================================================================

===TRANSACCIONES============================================================
    items     itemsetID
[1] {X,C,N,B} 1
[2] {X,C,B,T} 2
[3] {X,C,N}   3
[4] {X,N,B,T} 4
[5] {X,C,B}   5
[6] {N}       6
[7] {X,C,B}   7
[8] {T,A}     8
============================================================================

Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.9    0.1    1 none FALSE            TRUE       5     0.4      1     10  rules FALSE

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

Absolute minimum support count: 3

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[6 item(s), 8 transaction(s)] done [0.00s].
sorting and recoding items ... [4 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [3 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

```
===ASOCIACIONES============================================================
    lhs      rhs support confidence lift     count
[1] {C}   => {X} 0.625   1          1.333333 5
[2] {B}   => {X} 0.625   1          1.333333 5
[3] {C,B} => {X} 0.500   1          1.333333 4
============================================================================
```

### 2.1.3. Conclusiones

Una vez **ejecutado el algoritmo** podemos ver que solo hay tres asociaciones que superan el umbral de soporte y confianza fijados inicialemente a 40 % y 90 % respectivamente. Por tanto de la muestra inicial **podemos deducir** que:

- Cuando se compra el extra de Control de Velocidad se compra Faros de Xenon con una confianza del 100 %.
- Cuando se compra el extra de *Bluetooth* se compra Faros de Xenon con una confianza del 100 %.
- Cuando se compra los extras de Control de Velocidad y *Bluetooth* se compra Faros de Xenon con una confianza del 100 %.

Además, el ***lift*** en todas las asociaciones es mayor que uno por lo que podemos afirmar que al comprar los productos de la izquierda **aumenta la probabilidad** de comprar los productos de la derecha.
Para demostrar la solución obtenida por `R`, vamos a resolver el problema anterior aplicando los pasos del algoritmo visto en teoría.

### 2.1.4. Planteamiento del problema

**Paso A. Identificación de las asociaciones frecuentes. Cálculo del soporte** El primer paso consistirá en **analizar los sucesos elementales**, calculando su soporte y eliminando aquellos que no superen dicho umbral establecido. Recordemos que el soporte deberá ser **igual o superior** al 40 % y con una confianza **mayor o igual** al 80 %.

*Sucesos elementales*
*X, A, T, N, B, C*

| | Soporte |
|---|---|
| X | $6/8 = 0.75 = 75\,\%$ |
| A | $1/8 = 0.125 = 12.5\,\%$ |
| T | $3/8 = 0.375 = 37.5\,\%$ |
| N | $4/8 = 0.50 = 50\,\%$ |
| B | $5/8 = 0.625 = 62.5\,\%$ |
| C | $5/8 = 0.625 = 62.5\,\%$ |

Como podemos observar, **sólo los sucesos elementales *X, N, B y C* superar el umbral de soporte mínimo**. Como consecuencia, no estudiaremos el resto de sucesos elementales. De esta forma, reducimos el número de sucesos elementales a cuatro:

**X,N,B y C**

A continuación, aplicamos el **subpaso 2**, en el que comenzaremos con conjuntos de sucesos de dos dimensiones y terminaremos cuando no sea posible identificar una dimensión en la que haya un soporte **mayor o igual que el umbral**, mediante el algoritmo *apriori-gen*. Comenzamos con la **dimensión 2**: Aplicamos el método

$$F_{2-1} \times F_{2-1} = F_1 \times F_1$$

.

$$a_i = b_i$$

Para *i=1,2,. . . ,k-2* primeros,es decir, en este caso tendremos k-2 = 2-2 = 0. Esto quiere decir que deben coincidir los k-2 primeros. Tiene sentido ya que los sucesos elementales son únicos y no se pueden repetir.

$$a_{k-1} \neq b_{k-1}$$

Es decir, en este caso tendremos k-1 = 2-1 = 1. Esto quiere decir que no deben coincidir los elementos

$$a_1 \neq b_1$$

Entonces, siguiendo estas condiciones los conjuntos de dos dimensiones quedan de la siguiente forma:

**{X,N}, {X,B}, {X,C}, {N,B}, {N,C}, {B,C}**

Una vez obtenidos todas las posibles combinaciones de sucesos de dos dimensiones, **calculamos su soporte**:

| | Soporte |
|---|---|
| {X,N} | $3/8 = 0.375 = 37.5\,\%$ |
| {X,B} | $5/8 = 0.625 = 62.5\,\%$ |
| {X,C} | $5/8 = 0.625 = 62.5\,\%$ |
| {N,B} | $2/8 = 0.25 = 25\,\%$ |
| {N,C} | $2/8 = 0.25 = 25\,\%$ |
| {B,C} | $4/8 = 0.50 = 50\,\%$ |

Como podemos observar, solo tres sucesos logran superar el umbral de soporte: **{X,B}, {X,C}, {B,C}**

Continuamos con **dimensión 3**: Aplicamos el método

$$F_{3-1} \times F_{3-1} = F_2 \times F_2$$

$$a_i = b_i$$

Para *i=1,2,. . . ,k-2* primeros,es decir, en este caso tendremos k-2 = 3-2 = 1. Esto quiere decir que debe coincidir el primer elemento.

$$a_{k-1} \neq b_{k-1}$$

Es decir, en este caso tendremos k-1 = 3-1 = 2. Esto quiere decir que no deben coincidir los elementos

$$a_2 \neq b_2; a_1 \neq b_1$$

Entonces, siguiendo estas condiciones el único conjunto posible de tres dimensiones queda de la siguiente forma:

**{X,B,C}**

Una vez obtenidos todas las posibles combinaciones de sucesos de tres dimensiones, **calculamos su soporte**:

**Soporte {X,B,C} = 4/8 = 0.50 = 50 %**

Hemos llegado a la máxima dimensión. Por tanto, los sucesos candidatos de 2 y 3 dimensiones que superan el soporte establecido son:

**{X,B}, {X,C}, {B,C}, {X,B,C}**

A continuación, cambiamos cada suceso elemental por un valor numérico:

- X: 1
- A: 2
- T: 3
- N: 4
- B: 5
- C: 6

De esta manera, los sucesos candidatos nos quedan de la siguiente forma:

**{1,5}, {1,6}, {5,6}, {1,5,6}**

Una vez asignado un valor a cada suceso elemental, aplicaremos la **función de partición**:

$$h(p) = p \mod 3$$

Donde **p** es el valor correspondiente a cada suceso elemental. Como la función de partición es **mod 3**, recordemos que tendremos un total de 3 nodos que se crearán de la siguiente forma:

- *h(1) = 1 mod 3 = 1*
- *h(2) = 2 mod 3 = 2*
- *h(3) = 3 mod 3 = 0 ó 3*
- *h(4) = 4 mod 3 = 1*
- *h(5) = 5 mod 3 = 2*
- *h(6) = 6 mod 3 = 0 ó 3*

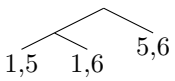
Como podemos comprobar, la función de partición nos ha dado valores comprendidos entre 1 y 3. Esto nos indica a qué nodo va cada uno de los sucesos elementales. Esos nodos quedarán de la siguiente forma:

- *Nodo 1 = {1,4}*
- *Nodo 2 = {2,5}*
- *Nodo 3 = {3,6}*

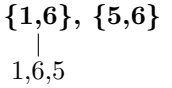
Como podemos observar, los nodos se organizan de **izquierda a derecha**. Comenzamos con los sucesos de **dos dimensiones**:

**{1,5}, {1,6}, {5,6}**

En primer lugar, comprobamos el primer valor de cada suceso. En este caso, los dos primeros primeros sucesos se situarán en el primer nodo del árbol (ya que ambos empiezan por 1), mientras que el tercer suceso se situará en el segundo nodo. Por otro lado, *{1,5}* se situará a la izquierda de *{1,6}*, ya que el 5 pertenece a la clase 2 mientras que 6 pertenece a la clase 3:



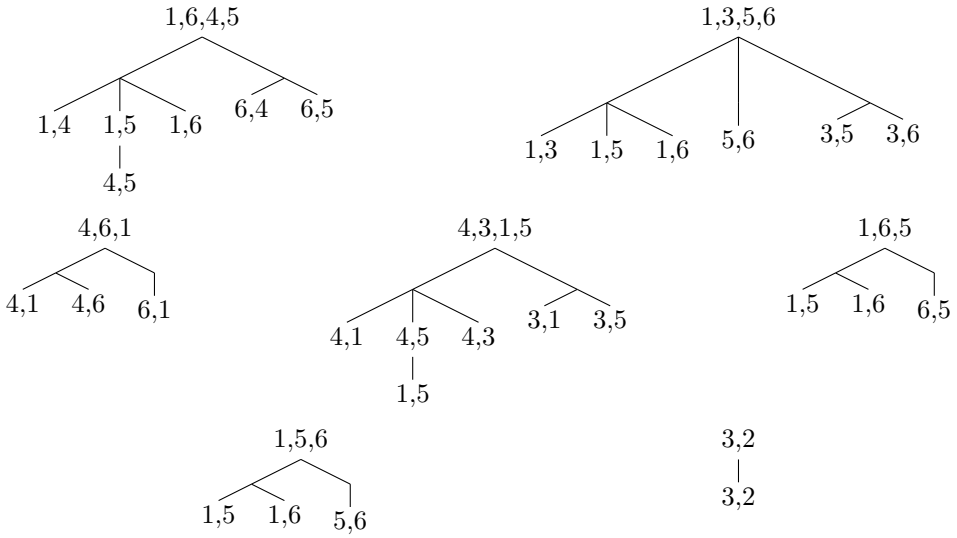
Realizamos el mismo proceso para los sucesos de **tres dimensiones**:



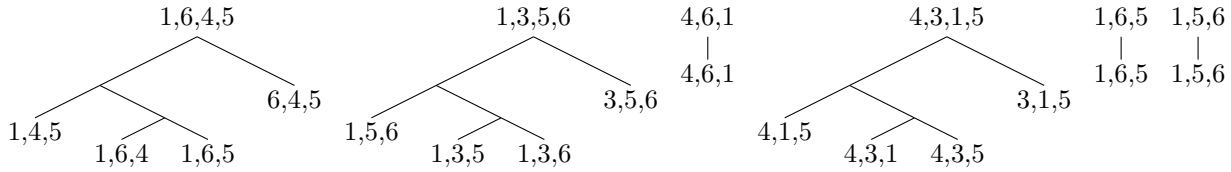

En este caso sólo tenemos un suceso candidato. De esta forma ya tendríamos las particiones de los sucesos candidatos. A continuación, **realizamos una partición de los sucesos de la muestra, utilizando el mismo *hash tree***. Si pasamos los sucesos elementales a valores numéricos, las muestras quedarán de la siguiente forma:

- $\{X,C,N,B\} = \{1,6,4,5\}$
- $\{X,T,B,C\} = \{1,3,5,6\}$
- $\{N,C,X\} = \{4,6,1\}$
- $\{N,T,X,B\} = \{4,3,1,5\}$
- $\{X,C,B\} = \{1,6,5\}$
- $\{N\} = \{4\}$
- $\{X,B,C\} = \{1,5,6\}$
- $\{T,A\} = \{3,2\}$

**Construimos las particiones para sucesos de 2 dimensiones**



Si nos fijamos no hemos puesto el suceso {4} ya que es inferior a la dimensión establecida. **Ahora, construimos las particiones para subconjuntos de 3 dimensiones**:




Si nos fijamos no hemos puesto el suceso {1 2} ni el {4} ya que es inferior a la dimensión establecida. Y con esto ya tendríamos las particiones de las muestras. A continuación, **contamos el número de aparaciones de cada suceso candidato**: **Para subconjuntos de 2 dimensiones**

- $\{1,5\} = \{X,B\} = 5$, luego su soporte es de $5/8 = 0.625 = 62.5\,\%$
- $\{1,6\} = \{X,C\} = 4$, luego su soporte es de $4/8 = 0.50 = 50\,\%$
- $\{5,6\} = \{C,B\} = 4$, luego su soporte es de $4/8 = 0.50 = 50\,\%$

**Para subconjuntos de 3 dimensiones**

- $\{1,6,5\} = \{X,C,B\} = 4$, luego su soporte es de $4/8 = 0.50 = 50\,\%$

En todos los sucesos candidatos, el soporte es **mayor o igual** al 40 %, luego todos son aptos para continuar con su estudio.

**Paso B. Identificación de las asociaciones de confianza. Cálculo de la confianza** El siguiente paso consistirá en **analizar los sucesos candidatos** que hayan superado el umbral de soporte. Para ello, **identificaremos las asociaciones:**

$$2^k - 2$$

- Para $k = 2$ dimensiones, tendremos **2 asociaciones por cada suceso candidato**.
- Para $k = 3$ dimensiones, tendremos **6 asociaciones por cada suceso candidato**.

**Para sucesos candidatos de 2 dimensiones**

$$\{X\} \rightarrow \{B\}$$

$$\{X\} \rightarrow \{C\}$$

$$\{C\} \rightarrow \{B\}$$

$$\{B\} \rightarrow \{X\}$$

$$\{C\} \rightarrow \{X\}$$

$$\{B\} \rightarrow \{C\}$$

**Para sucesos candidatos de 3 dimensiones**

$$\{X, C\} \rightarrow \{B\}$$

$$\{X, B\} \rightarrow \{C\}$$

$$\{C, B\} \rightarrow \{X\}$$

$$\{B\} \rightarrow \{X, C\}$$

$$\{C\} \rightarrow \{B, X\}$$

$$\{X\} \rightarrow \{C, B\}$$

A continuación, aplicamos el siguiente teorema:
**Comenzamos con $B = \{X,C,B\}$**:

- Para $A = \{X,C\}$.

  $$\{X, C\} \rightarrow \{X, B, C\} - \{X, C\} = \{X, C\} \rightarrow \{B\}$$

  Calculamos su confianza:

  $$c(\{X, C\} \rightarrow \{B\}) = \frac{4}{5} = 0{,}8 < 0{,}9$$

  Al no superar el umbral, cualquier asociación

  $$A' \rightarrow B - A'$$

  no lo alcanzarán.

- Para $A = \{X,B\}$.

  $$\{X, B\} \rightarrow \{X, B, C\} - \{X, B\} = \{X, B\} \rightarrow \{C\}$$

  Calculamos su confianza:

  $$c(\{X, B\} \rightarrow \{C\}) = \frac{4}{5} = 0{,}8 < 0{,}9$$

  Al no superar el umbral, cualquier asociación

  $$A' \rightarrow B - A'$$

  no lo alcanzarán.

- Para $A = \{C,B\}$.

  $$\{C, B\} \rightarrow \{X, B, C\} - \{C, B\} = \{C, B\} \rightarrow \{X\}$$

  Calculamos su confianza:

  $$c(\{C, B\} \rightarrow \{X\}) = \frac{4}{4} = 1 > 0{,}9$$

  En este caso, no podemos aplicar el teorema: Si Aés cada uno de los subconjuntos de A, entonces A'= $\{C\}$.

  $$\{C\} \rightarrow \{X, C, B\} - \{C\} = \{C\} \rightarrow \{X, B\}$$

Calculamos su confianza:

$$c(\{C\} \to \{X, B\}) = \frac{4}{5} = 0{,}8 < 0{,}9$$

A'= $\{B\}$.

$$\{B\} \to \{X, C, B\} - \{B\} = \{B\} \to \{X, C\}$$

Calculamos su confianza:

$$c(\{B\} \to \{X, C\}) = \frac{4}{5} = 0{,}8 < 0{,}9$$

**Cogemos el siguiente suceso candidato: $B = \{C,B\}$**:

- Para $A = \{C\}$.

  $$\{C\} \to \{C, B\} - \{C\} = \{C\} \to \{B\}$$

  Calculamos su confianza:

  $$c(\{C\} \to \{B\}) = \frac{4}{5} = 0{,}8 < 0{,}9$$

- Para $A = \{B\}$.

  $$\{B\} \to \{C, B\} - \{B\} = \{B\} \to \{C\}$$

  Calculamos su confianza:

  $$c(\{B\} \to \{C\}) = \frac{4}{5} = 0{,}8 < 0{,}9$$

**Volvemos a coger el siguiente suceso candidato: $B = \{X,C\}$**:

- Para $A = \{X\}$.

  $$\{X\} \to \{X, C\} - \{X\} = \{X\} \to \{C\}$$

  Calculamos su confianza:

  $$c(\{X\} \to \{C\}) = \frac{5}{6} = 0{,}83 < 0{,}9$$

- Para $A = \{C\}$.

  $$\{C\} \to \{X, C\} - \{C\} = \{C\} \to \{X\}$$

  Calculamos su confianza:

  $$c(\{C\} \to \{X\}) = \frac{5}{5} = 1 > 0{,}9$$

**Y cojemos el último suceso candidato: $B = \{X,B\}$**:

- Para $A = \{X\}$.

  $$\{X\} \to \{X, B\} - \{X\} = \{X\} \to \{B\}$$

  Calculamos su confianza:

  $$c(\{X\} \to \{B\}) = \frac{5}{6} = 0{,}83 < 0{,}9$$

  .

- Para $A = \{B\}$.

  $$\{B\} \to \{X, B\} - \{B\} = \{B\} \to \{X\}$$

  Calculamos su confianza:

  $$c(\{B\} \to \{X\}) = \frac{5}{5} = 1 > 0{,}9$$

Para concluir: **Las asociaciones con un soporte mayor o igual que el 40 % y una confianza mayor o igual que el 90 % son:**

$$\{C, B\} \to \{X\}$$
$$\{C\} \to \{X\}$$
$$\{B\} \to \{X\}$$

## 2.2. Apartado 2.2

Este apartado consistirá en el análisis de asociación con `R` utilizando **datos de importaciones y exportaciones realizadas en la India entre los años 2010 y 2018**[2] **en formato *.xlsx***. Para la realización de este apartado, utilizaremos las siguiente librerías:

- *package(openxlsx)* para la lectura de ficheros *.xlsx* de gran tamaño[3]. La diferencia con respecto al paquete *XLConnect* es que elimina la dependencia de *Java* para la lectura de ficheros.
- *package(arules)* para la ejecución del algoritmo *apriori*.
- *package(dplyr)*, el cual dispone de una gramática propia para la manipulación de datos, similares a una consulta `SQL`.
- *package(plyr)* para la partición de grandes volúmenes de datos, permitiendo trabajar con pequeños subconjuntos [4].
- *package(arulesViz)*. Se trata de una extensión del paquete *arules* que ofrece varias técnicas de visualización para conjuntos de asociaciones [5].

Inicialmenete, comenzamos instalando y cargando todos los datos:

```
> #Instalamos y cargamos los paquetes
> if(!require(openxlsx)){
+     install.packages("openxlsx")
+     require(openxlsx)
+ }
> if(!require(arules)){
+     install.packages("arules")
+     require(arules)
+ }
> if(!require(dplyr)){
+     install.packages("dplyr")
+     require(dplyr)
+ }
> if(!require(plyr))
+ {
+     install.packages("plyr")
+     require(plyr)
+ }
> if(!require(arulesViz)){
+     install.packages("arulesViz")
+     require(arulesViz)
+ }
```

[2] https://www.kaggle.com/lakshyaag/india-trade-data
[3] https://cran.r-project.org/web/packages/openxlsx/index.html
[4] https://cran.r-project.org/web/packages/plyr/index.html
[5] https://cran.r-project.org/web/packages/arulesViz/vignettes/arulesViz.pdf

Comenzamos con la lectura del fichero *.xlsx*. Para ello, *openxlsx* dispone de una función denominada *read.xlsx*:

- nombre del archivo.
- *sheet*: número de hoja a leer.
- *startRow*: número de fila de comienzo del fichero.
- *endCol*: número de columna final del fichero.

```
> #Datos de exportaciones
> listado.exportaciones <- read.xlsx("2018-2010_export.xlsx", sheet = 1, startRow = 1, colNames = TRUE)
> nrow(listado.exportaciones)
```

```
[1] 137023
```

```
> #Datos de importaciones
> listado.importaciones <- read.xlsx("2018-2010_import.xlsx", sheet = 1, startRow = 1, colNames = TRUE)
> nrow(listado.importaciones)
```

```
[1] 93095
```

Debido al elevado número de filas que presenta el *dataset*, vamos a realizar una pequeña consulta sobre nuestro *dataframe* mediante la librería *dplyr*. Si recordamos de la práctica anterior, esta librería permite realizar consultas a un dataframe, similar a una consulta `SQL` [6], proporcionando los siguientes comandos para realizar consultas sobre un data frame:

- **select**: permite seleccionar un conjunto de columnas.
- **filter**: devuelve un conjunto de filas que cumplan una condición dada.
- **arrange**: permite reordenar las filas de un data frame.
- **rename**: permite renombrar variables en un data frame.
- **mutate**: permite añadir nuevas columnas o modificar columnas existentes.
- **head**: para obtener las primeras n filas.
- **summarise**: para calcular resúmenes estadísticos.
- **pipe**: se emplea para concatenar varias acciones.

En este caso, vamos a consultar **las 10 primeras filas**:

```
> #pipe = %>%, muy importante para concatenar varias acciones sobre un dataframe
> #Equivalente en SQL a:
> #SELECT Commodity FROM listado.exportaciones LIMIT 10
> listado.exportaciones %>% select(Commodity) %>% head(10)
```

```
                                                                                                                Commodity
1                                                                                                MEAT AND EDIBLE MEAT OFFAL.
2                                                                                FISH AND CRUSTACEANS, MOLLUSCS AND OTHER AQUATIC INVERTABRATES.
3    DAIRY PRODUCE; BIRDS' EGGS; NATURAL HONEY; EDIBLE PROD. OF ANIMAL ORIGIN, NOT ELSEWHERE SPEC. OR INCLUDED.
4                  LIVE TREES AND OTHER PLANTS; BULBS; ROOTS AND THE LIKE; CUT FLOWERS AND ORNAMENTAL FOLIAGE.
5                                                                EDIBLE VEGETABLES AND CERTAIN ROOTS AND TUBERS.
6                                                            EDIBLE FRUIT AND NUTS; PEEL OR CITRUS FRUIT OR MELONS.
7                                                                                  COFFEE, TEA, MATE AND SPICES.
8                                                                                                       CEREALS.
9                                                  PRODUCTS OF THE MILLING INDUSTRY; MALT; STARCHES; INULIN; WHEAT GLUTEN.
10 OIL SEEDS AND OLEA. FRUITS; MISC. GRAINS, SEEDS AND FRUIT; INDUSTRIAL OR MEDICINAL PLANTS; STRAW AND FODDER.
```

[6] https://rsanchezs.gitbooks.io/rprogramming/content/chapter9/dplyr.html

```
> #SELECT Commodity FROM listado.importaciones LIMIT 10
> listado.importaciones %>% select(Commodity) %>% head(10)
                                                                                                                  Commodity
1                                                                PRODUCTS OF ANIMAL ORIGIN, NOT ELSEWHERE SPECIFIED OR INCLUDED.
2                                                                                EDIBLE VEGETABLES AND CERTAIN ROOTS AND TUBERS.
3                                                                          EDIBLE FRUIT AND NUTS; PEEL OR CITRUS FRUIT OR MELONS.
4                                                                                                  COFFEE, TEA, MATE AND SPICES.
5                                                              PRODUCTS OF THE MILLING INDUSTRY; MALT; STARCHES; INULIN; WHEAT GLUTEN.
6  OIL SEEDS AND OLEA. FRUITS; MISC. GRAINS, SEEDS AND FRUIT; INDUSTRIAL OR MEDICINAL PLANTS; STRAW AND FODDER.
7                                                                        LAC; GUMS, RESINS AND OTHER VEGETABLE SAPS AND EXTRACTS.
8                                                             PREPARATIONS OF VEGETABLES, FRUIT, NUTS OR OTHER PARTS OF PLANTS.
9                                                           SALT; SULPHUR; EARTHS AND STONE; PLASTERING MATERIALS, LIME AND CEMENT.
10        MINERAL FUELS, MINERAL OILS AND PRODUCTS OF THEIR DISTILLATION; BITUMINOUS SUBSTANCES; MINERAL WAXES.
```

Analicemos las columnas de ambos ficheros:

- **HSCode**: id único para cada producto de importacion/exportacion.
- **Commodity**: nombre del producto.
- **value**: precio de cada producto.
- **country**: país de importación/exportación.
- **year**: año de importación/exportación.

Una vez importados los datos, comprobamos si existen posibles filas que contengan valores a *NA*:

```
> #Para los datos de exportacion
> sum(is.na(listado.exportaciones))

[1] 14038

> #Para los datos de importacion
> sum(is.na(listado.importaciones))

[1] 14027
```

Dado que existen filas con valores a *NA*, debemos conservar únicamente aquellas **filas completas**, es decir, sin valores a *NA*. Para ello, utilizaremos la función *complete.cases*, que devolverá un *booleano* por cada fila, indicando si se trata o no de una fila completa [7]:

```
> #Para datos de exportacion/importacion, nos quedamos con aquellas filas que no contengan
> #columnas a NA. para ello, utilizaremos la función complete.cases() que
> #filtran aquellas filas "completas"
> listado.exportacionesv2 <- listado.exportaciones[complete.cases(listado.exportaciones), ]
> sum(is.na(listado.exportacionesv2)) #0, luego ya no existen valores a NA

[1] 0

> listado.importacionesv2 <- listado.importaciones[complete.cases(listado.importaciones), ]
> sum(is.na(listado.importacionesv2)) #0, luego ya no existen valores a NA

[1] 0
```

Una vez eliminadas las filas con valores a *NA*, debemos **transformar nuestro fichero Excel a transacciones**. Para ello, en lugar de crear una matriz binaria como en el apartado 1, utilizaremos la función *ddply* [8], disponible en el paquete *dplyr*. Esta función permite **aplicar una función a un subconjunto del *data frame***. En nuestro caso, queremos juntar los datos mediante la función *paste* agrupados por la columna *country*:

[7] https://www.rdocumentation.org/packages/stats/versions/3.6.1/topics/complete.cases
[8] https://www.rdocumentation.org/packages/plyr/versions/1.8.4/topics/ddply

```
> #Mediante el paquete plyr, agrupamos los productos por pais
> #Utilizando la funcion paste, juntamos la columna Commodity de
> #aquellas exportaciones/importaciones del mismo pais
> transactionData.exportaciones <- ddply(listado.exportacionesv2,c("country"),
+ function(df1)paste(df1$Commodity, collapse = ","))
> transactionData.importaciones <- ddply(listado.importacionesv2,c("country"),
+ function(df1)paste(df1$Commodity, collapse = ","))
```

Una vez agrupados los datos en transacciones, eliminaremos la columna *country*, pues ya no va a ser necesaria, además de renombrar la columna de transacciones:

```
> #Eliminamos la columna country
> transactionData.exportaciones$country <- NULL
> transactionData.importaciones$country <- NULL
> #Renombramos la columna de transacciones
> #Por defecto, se llama V1
> colnames(transactionData.exportaciones) <- c("exportaciones")
> colnames(transactionData.importaciones) <- c("importaciones")
```

A continuación, exportamos los *data frames* resultantes a formato *csv*. Para ello, utilizaremos la función *write.csv*:

```
> #Guardamos la transacciones resultantes a csv
> #Quote: eliminamos posibles comillas dobles que aparezcan
> #Row.names: si queremos enumerar o no cada fila
> write.csv(transactionData.exportaciones,"exportaciones.csv", quote = FALSE,
+ row.names = FALSE)
> write.csv(transactionData.importaciones,"importaciones.csv", quote = FALSE,
+ row.names = FALSE)
```

Una vez exportados ambos *data frames*, mediante la librería *arules*, realizamos la lectura del fichero de transacciones. Para ello, utilizaremos la función *read.transactions*. Por lo general, existen dos formas de leer un fichero de transacciones:

- *basket*: cuando cada transacción incluye una *cesta* de sucesos, donde cada elemento está separado con el caracter que indiquemos en *read.transactions*.
- *single*: cuando cada transacción contiene un único elemento, o parejas *id-producto*.

En nuestro caso, se trata de un fichero de tipo *basket*:

```
> #Para datos de exportaciones
> tr.exportaciones <- read.transactions('exportaciones.csv', format = 'basket', sep=',')
> #Para datos de importaciones
> tr.importaciones <- read.transactions('importaciones.csv', format = 'basket', sep=',')
```

Tras importar las transacciones, ejecutamos los algoritmos de asociación: **Comenzamos con el algoritmo *apriori***:

```
> #Para los datos de exportacion
> #Ejectuamos el algoritmo apriori con un 77% de soporte y un 90% de confianza
> association.rules.exportaciones <- apriori(tr.exportaciones, parameter = list(supp=0.77, conf=0.9))
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.9    0.1    1 none FALSE            TRUE       5    0.77      1     10  rules FALSE

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

Absolute minimum support count: 191

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[985 item(s), 249 transaction(s)] done [0.01s].
sorting and recoding items ... [7 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [18 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

```
> #Para los datos de importacion
> #Ejectuamos el algoritmo apriori con un 86% de soporte y un 90% de confianza
> association.rules.importaciones <- apriori(tr.importaciones, parameter = list(supp=0.86, conf=0.9))

Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.9    0.1    1 none FALSE            TRUE       5    0.86      1     10  rules FALSE

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

Absolute minimum support count: 208

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[528 item(s), 242 transaction(s)] done [0.00s].
sorting and recoding items ... [6 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [18 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].

> #Para los datos de exportacion
> inspect(association.rules.exportaciones)

     lhs
[1]  {NUCLEAR REACTORS}
[2]  {BOILERS}
[3]  {NUCLEAR REACTORS}
[4]  {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}
[5]  {BOILERS}
[6]  {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}
```

```
[7]  {AND PARTS.}
[8]  {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS}
[9]  {AND PARTS.}
[10] {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}
[11] {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS}
[12] {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}
[13] {BOILERS,NUCLEAR REACTORS}
[14] {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.,NUCLEAR REACTORS}
[15] {BOILERS,MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}
[16] {AND PARTS.,ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS}
[17] {AND PARTS.,TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}
[18] {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS,TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}
     rhs                                                                                                                                  support   confidence lift     count
[1]  {BOILERS}                                                                                                                            0.7710843 1          1.296875 192
[2]  {NUCLEAR REACTORS}                                                                                                                   0.7710843 1          1.296875 192
[3]  {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}                                                                                0.7710843 1          1.296875 192
[4]  {NUCLEAR REACTORS}                                                                                                                   0.7710843 1          1.296875 192
[5]  {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}                                                                                0.7710843 1          1.296875 192
[6]  {BOILERS}                                                                                                                            0.7710843 1          1.296875 192
[7]  {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS}                                              0.7991968 1          1.251256 199
[8]  {AND PARTS.}                                                                                                                         0.7991968 1          1.251256 199
[9]  {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}                                                                               0.7991968 1          1.251256 199
[10] {AND PARTS.}                                                                                                                         0.7991968 1          1.251256 199
[11] {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}                                                                               0.7991968 1          1.251256 199
[12] {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS}                                              0.7991968 1          1.251256 199
[13] {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}                                                                                0.7710843 1          1.296875 192
[14] {BOILERS}                                                                                                                            0.7710843 1          1.296875 192
[15] {NUCLEAR REACTORS}                                                                                                                   0.7710843 1          1.296875 192
[16] {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}                                                                               0.7991968 1          1.251256 199
[17] {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS}                                              0.7991968 1          1.251256 199
[18] {AND PARTS.}                                                                                                                         0.7991968 1          1.251256 199

> #Para los datos de importacion
> inspect(association.rules.importaciones)

     lhs
[1]  {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}
[2]  {NUCLEAR REACTORS}
[3]  {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}
[4]  {BOILERS}
[5]  {NUCLEAR REACTORS}
[6]  {BOILERS}
[7]  {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS}
[8]  {AND PARTS.}
[9]  {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS}
[10] {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}
[11] {AND PARTS.}
[12] {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}
[13] {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.,NUCLEAR REACTORS}
[14] {BOILERS,MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}
[15] {BOILERS,NUCLEAR REACTORS}
[16] {AND PARTS.,ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS}
[17] {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS,TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}
[18] {AND PARTS.,TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}
     rhs                                                                                                                                  support   confidence lift     count
[1]  {NUCLEAR REACTORS}                                                                                                                   0.8636364 1          1.157895 209
[2]  {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}                                                                                0.8636364 1          1.157895 209
[3]  {BOILERS}                                                                                                                            0.8636364 1          1.157895 209
[4]  {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}                                                                                0.8636364 1          1.157895 209
[5]  {BOILERS}                                                                                                                            0.8636364 1          1.157895 209
[6]  {NUCLEAR REACTORS}                                                                                                                   0.8636364 1          1.157895 209
[7]  {AND PARTS.}                                                                                                                         0.8677686 1          1.152381 210
[8]  {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS}                                              0.8677686 1          1.152381 210
[9]  {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}                                                                               0.8677686 1          1.152381 210
[10] {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS}                                              0.8677686 1          1.152381 210
[11] {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}                                                                               0.8677686 1          1.152381 210
[12] {AND PARTS.}                                                                                                                         0.8677686 1          1.152381 210
[13] {BOILERS}                                                                                                                            0.8636364 1          1.157895 209
[14] {NUCLEAR REACTORS}                                                                                                                   0.8636364 1          1.157895 209
[15] {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}                                                                                0.8636364 1          1.157895 209
[16] {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}                                                                               0.8677686 1          1.152381 210
[17] {AND PARTS.}                                                                                                                         0.8677686 1          1.152381 210
[18] {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS}                                              0.8677686 1          1.152381 210
```

A continuación, **emplearemos el algoritmo *ECLAT* para el cáclulo de asociación**, disponible en el paquete *arules*. El algoritmo *ECLAT* (*Equivalence Class Clustering and bottom-up Lattice Traversal*) junto con *apriori* se tratan de los algoritmos de asociación más populares. A diferencia de *a priori*, *ECLAT* trabaja verticalmente con los conjuntos de sucesos, de forma similar a una búsqueda en profundidad. Supongamos que tenemos la siguiente matriz binaria, en la que debemos aplicar asociación **con un soporte mínimo del 20 %**:

| | Bread | Butter | Milk | Coke | Jam |
|---|---|---|---|---|---|
| T1 | 1 | 1 | 0 | 0 | 1 |
| T2 | 0 | 1 | 0 | 1 | 0 |
| T3 | 0 | 1 | 1 | 0 | 0 |
| T4 | 1 | 1 | 0 | 1 | 0 |
| T5 | 1 | 0 | 1 | 0 | 0 |
| T6 | 0 | 1 | 1 | 0 | 0 |
| T7 | 1 | 0 | 1 | 0 | 0 |
| T8 | 1 | 1 | 1 | 0 | 1 |
| T9 | 1 | 1 | 1 | 0 | 0 |

En la primera iteración **agrupamos las transacciones por cada uno de los sucesos elementales**:

| Suceso elemental | Conjunto de sucesos |
|---|---|
| Bread | {T1, T4, T5, T7, T8, T9} |
| Butter | {T1, T2, T3, T4, T6, T8, T9} |
| Milk | {T3, T5, T6, T7, T8, T9} |
| Coke | {T2, T4} |
| Jam | {T1, T8} |

En la segunda iteración, con los sucesos elementales **creamos parejas de sucesos, agrupando de nuevo las transacciones con los nuevos sucesos**, y así de forma recursiva hasta que no se puedan formar conjuntos de sucesos[9]:

| Suceso elemental | Conjunto de sucesos |
|---|---|
| {Bread, Butter} | {T1, T4, T8, T9} |
| {Bread, Milk} | {T5, T7, T8, T9} |
| {Bread, Coke} | {T4} |
| {Bread, Jam} | {T1,T8} |
| {Butter, Milk} | {T3, T6, T8, T9} |
| {Butter, Coke} | {T2, T4} |
| {Butter, Jam} | {T1, T8} |
| {Milk, Jam} | {T8} |

**Para conjuntos de sucesos de tres dimensiones**:

| Suceso elemental | Conjunto de sucesos |
|---|---|
| {Bread, Butter, Milk} | {T8, T9} |
| {Bread, Butter, Jam} | {T1, T8} |

**Para conjuntos de sucesos de cuatro dimensiones**:

| Suceso elemental | Conjunto de sucesos |
|---|---|
| {Bread, Butter, Milk, Jam} | {T8} |

[9]Para el conjunto de sucesos {Coke, Jam}, por ejemplo, no aparecen conjuntamente en ninguna de las transacciones anteriores, por lo que no lo incluimos.

Una vez tengamos nuestro conjunto de **sucesos candidatos**, filtramos aquellos conjuntos con un soporte **mayor o igual al 20 %**:

| Soporte mayor o igual a 20 |
|---|
| {Bread, Butter} |
| {Bread, Milk} |
| {Bread, Jam} |
| {Butter, Milk} |
| {Butter, Coke} |
| {Butter, Jam} |
| {Bread, Butter, Milk} |
| {Bread, Butter, Jam} |
| {Bread, Butter, Milk, Jam} |

A partir de los sucesos anteriores, **podemos establecer las reglas de asociación, creando todas las posibles combinaciones con cada suceso, sin repetir asociaciones**:

| Suceso 1 | Suceso 2 |
|---|---|
| {Bread} | {Butter} |
| {Bread} | {Milk} |
| {Bread} | {Jam} |
| {Butter} | {Milk} |
| {Butter} | {Coke} |
| {Butter} | {Jam} |
| {Bread, Butter} | {Milk} |
| {Bread, Butter} | {Jam} |

**Una vez explicado el funcionamiento del algoritmo, mediante la función *eclat* del paquete *arules* aplicaremos el algoritmo con los datos de exportación e importación anteriores**:

```
> #Para datos de exportacion
> #Ejectuamos el algoritmo apriori con un 77% de soporte y un 90% de confianza
> #Maxlen indica el maximo numero de transacciones que deseamos obtener
> itemsets.exportacion <- eclat(tr.exportaciones, parameter = list(supp = 0.77, maxlen = 20))

Eclat

parameter specification:
 tidLists support minlen maxlen            target   ext
    FALSE    0.77      1     20 frequent itemsets FALSE

algorithmic control:
 sparse sort verbose
      7   -2    TRUE

Absolute minimum support count: 191

create itemset ...
set transactions ...[985 item(s), 249 transaction(s)] done [0.01s].
sorting and recoding items ... [7 item(s)] done [0.00s].
creating bit matrix ... [7 row(s), 249 column(s)] done [0.00s].
writing  ... [15 set(s)] done [0.00s].
Creating S4 object  ... done [0.00s].

> #Para indicar la confianza, utilizamos la funcion ruleInduction()
> rules.exportacion <- ruleInduction(itemsets.exportacion, tr.exportaciones, confidence = 0.9)
> inspect(rules.exportacion)

     lhs
[1]  {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.,NUCLEAR REACTORS}
[2]  {BOILERS,NUCLEAR REACTORS}
```

```
[3]  {BOILERS,MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}
[4]  {NUCLEAR REACTORS}
[5]  {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}
[6]  {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}
[7]  {BOILERS}
[8]  {NUCLEAR REACTORS}
[9]  {BOILERS}
[10] {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS,TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}
[11] {AND PARTS.,TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}
[12] {AND PARTS.,ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS}
[13] {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}
[14] {AND PARTS.}
[15] {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}
[16] {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS}
[17] {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS}
[18] {AND PARTS.}
     rhs                                                                                  support   confidence lift     itemset
[1]  {BOILERS}                                                                            0.7710843 1          1.296875 1
[2]  {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}                                0.7710843 1          1.296875 1
[3]  {NUCLEAR REACTORS}                                                                   0.7710843 1          1.296875 1
[4]  {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}                                0.7710843 1          1.296875 2
[5]  {NUCLEAR REACTORS}                                                                   0.7710843 1          1.296875 2
[6]  {BOILERS}                                                                            0.7710843 1          1.296875 3
[7]  {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}                                0.7710843 1          1.296875 3
[8]  {BOILERS}                                                                            0.7710843 1          1.296875 4
[9]  {NUCLEAR REACTORS}                                                                   0.7710843 1          1.296875 4
[10] {AND PARTS.}                                                                         0.7991968 1          1.251256 5
[11] {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS} 0.7991968 1          1.251256 5
[12] {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}                               0.7991968 1          1.251256 5
[13] {AND PARTS.}                                                                         0.7991968 1          1.251256 6
[14] {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}                               0.7991968 1          1.251256 6
[15] {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS} 0.7991968 1          1.251256 7
[16] {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}                               0.7991968 1          1.251256 7
[17] {AND PARTS.}                                                                         0.7991968 1          1.251256 8
[18] {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS} 0.7991968 1          1.251256 8

> #Para datos de importacion
> #Ejectuamos el algoritmo apriori con un 86% de soporte y un 90% de confianza
> itemsets.importacion <- eclat(tr.importaciones, parameter = list(supp = 0.86, maxlen = 20))

Eclat

parameter specification:
 tidLists support minlen maxlen            target   ext
    FALSE    0.86      1     20 frequent itemsets FALSE

algorithmic control:
 sparse sort verbose
      7   -2    TRUE

Absolute minimum support count: 208

create itemset ...
set transactions ...[528 item(s), 242 transaction(s)] done [0.00s].
sorting and recoding items ... [6 item(s)] done [0.00s].
creating bit matrix ... [6 row(s), 242 column(s)] done [0.00s].
writing  ... [14 set(s)] done [0.00s].
Creating S4 object  ... done [0.00s].

> rules.importacion <- ruleInduction(itemsets.importacion, tr.importaciones, confidence = 0.9)
> inspect(rules.exportacion)

     lhs
[1]  {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.,NUCLEAR REACTORS}
[2]  {BOILERS,NUCLEAR REACTORS}
[3]  {BOILERS,MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}
[4]  {NUCLEAR REACTORS}
[5]  {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}
[6]  {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.}
[7]  {BOILERS}
[8]  {NUCLEAR REACTORS}
[9]  {BOILERS}
[10] {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS,TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}
[11] {AND PARTS.,TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}
```

```
[12] {AND PARTS.,ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS}
[13] {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}
[14] {AND PARTS.}
[15] {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS}
[16] {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS}
[17] {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS}
[18] {AND PARTS.}
```

| | rhs | support | confidence | lift | itemset |
|---|---|---|---|---|---|
| [1] | {BOILERS} | 0.7710843 | 1 | 1.296875 | 1 |
| [2] | {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.} | 0.7710843 | 1 | 1.296875 | 1 |
| [3] | {NUCLEAR REACTORS} | 0.7710843 | 1 | 1.296875 | 1 |
| [4] | {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.} | 0.7710843 | 1 | 1.296875 | 2 |
| [5] | {NUCLEAR REACTORS} | 0.7710843 | 1 | 1.296875 | 2 |
| [6] | {BOILERS} | 0.7710843 | 1 | 1.296875 | 3 |
| [7] | {MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.} | 0.7710843 | 1 | 1.296875 | 3 |
| [8] | {BOILERS} | 0.7710843 | 1 | 1.296875 | 4 |
| [9] | {NUCLEAR REACTORS} | 0.7710843 | 1 | 1.296875 | 4 |
| [10] | {AND PARTS.} | 0.7991968 | 1 | 1.251256 | 5 |
| [11] | {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS} | 0.7991968 | 1 | 1.251256 | 5 |
| [12] | {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS} | 0.7991968 | 1 | 1.251256 | 5 |
| [13] | {AND PARTS.} | 0.7991968 | 1 | 1.251256 | 6 |
| [14] | {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS} | 0.7991968 | 1 | 1.251256 | 6 |
| [15] | {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS} | 0.7991968 | 1 | 1.251256 | 7 |
| [16] | {TELEVISION IMAGE AND SOUND RECORDERS AND REPRODUCERS} | 0.7991968 | 1 | 1.251256 | 7 |
| [17] | {AND PARTS.} | 0.7991968 | 1 | 1.251256 | 8 |
| [18] | {ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS} | 0.7991968 | 1 | 1.251256 | 8 |

Por último, vamos a representar gráficamente las reglas de asociación. Para ello, utilizaremos el paquete *arulesViz*, que sobreescribe la función *plot* permitiendo la representación gráfica de reglas de asociación, utilizando el motor *graphViz*: [10]

[10] http://www.graphviz.org/


Figura 1: Plot
Nota: para visualizar mejor el grafo anterior se recomienda ejecutar el siguiente comando:
*plot(association.rules.exportaciones, method = "graph", engine = "htmlwidget").*
Al ejecutar se abrirá una extensión *HTML* con el grafo anterior, con la novedad de poder observar cada regla al acercar el ratón a cada nodo del grafo.