Ciencia de datos, práctica 2

Juan Casado Ballesteros, Samuel García Gonzalez, Iván Anaya Martín

October 18, 2019

Abstract

Para esta práctica vamos a investigar sobre los algoritmos de asociación. Utilizaremos apriori para analizar cestas de la compra y datos de vehículos. Posterioremente explicaremos como funciona el algoritmo de apriori y proporcionaremos un pequeño manual sobre cómo utilizarlo.

Para la parte final aplicaremos el algoritmo apriori sobre conjuntos de datos reales. También estudiaremos otros algoritmos que calculen la asociación y compararemos sus resultados respecto de los de apriori. Para estos otros algoritmos haremos también un manual sobre cómo utilizarlos.

Contents

1 Asociación sobre los datos de las cestas de la compra

Hemos introducido los datos de las cestas de la compra en un fichero .txt para evitar tener que escribirlos varias veces. Utilizamos funciones que hemos creado para automatizar la lectura del fichero y también para utilizar el algoritmo apriori. En estas funciones solo realizamos una lectura y una ejecición del algoritmo con los parámetros que nosotros hemos elegido. En el caso del fichero txt guardamos los elementos como listas de la compra en horizonal y los transformamos a como la función que llama a apriori espera recibirlos. En la función que llama a apriori configuramos como queremos utilizar el algoritmo de modo que no se nos impriman los detalles de su ejecución ni se calculen las asociaciones con conjuntos vacíos.

1.1 Cargar los datos de un .txt

Convierte la matriz a datos booleanos y la transpone, nos es más fácil escribir los datos en horizontal y transponer que no escribirlos ya como se espera que estén.

```
> readAprioriFile

function(file){
  muestra<-Matrix(as.matrix(read.table(file)), sparse=T)
  muestrangCMatrix<-as(muestra,"nsparseMatrix")
  t(muestrangCMatrix)
}
```

1.2 Llamar a apriori

Con minlen=2 indicamos que no queremos calcular asociaciones con el conjunto vacío y con verbose=F que nos queremos obtener el progreso del algoritmo.

```
> calapriori

function(matrix,soporte,confianza){
    transacciones<-as(matrix,"transactions")
    asociaciones <- apriori(transacciones,
        parameter=list(minlen=2, support=soporte,confidence=confianza),
        control=list(verbose=F))
    inspect(asociaciones)
    asociaciones
}
```

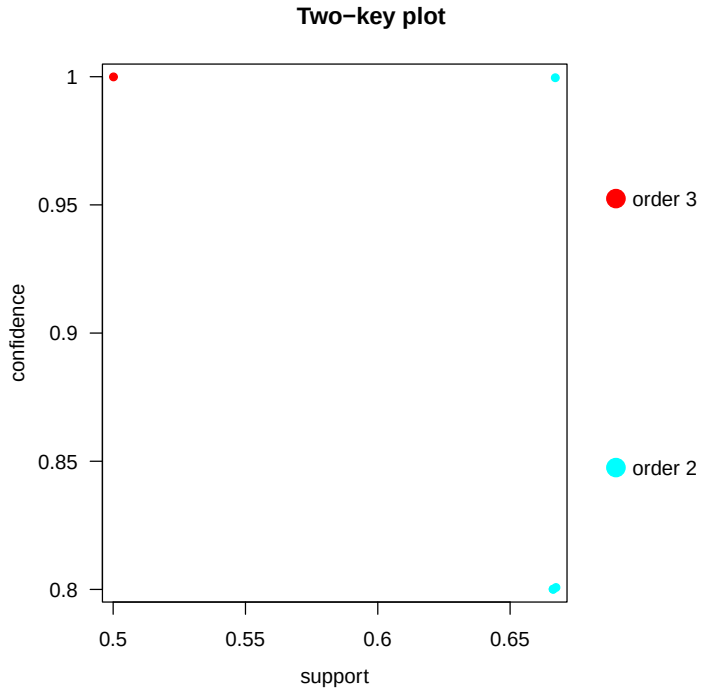
1.3 Calcular asociación

Calculamos la asociación con soporte 0.5 y confianza 0.8 para los datos de las cestas de la compra.

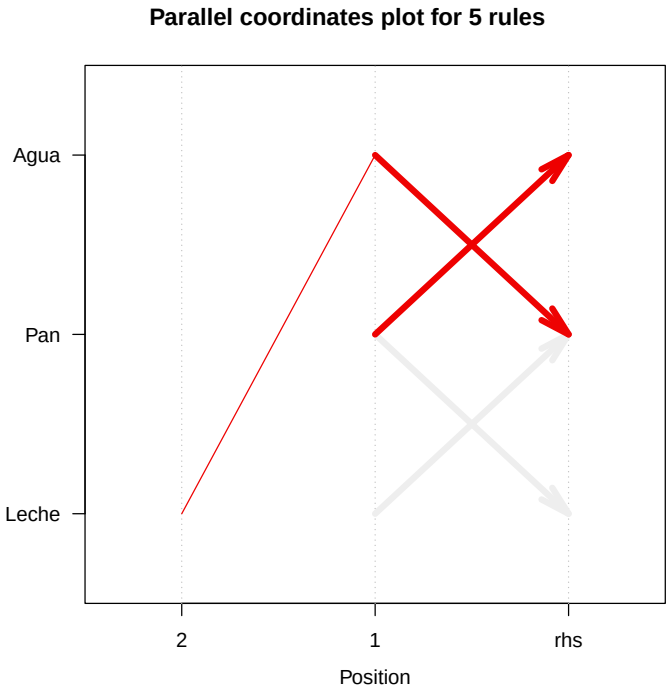
```
> shop_data = calapriori(readAprioriFile("datos1.txt"),0.5,0.8)
```

```
    lhs             rhs     support   confidence lift count
[1] {Agua}       => {Pan}   0.6666667 1.0        1.20 4
[2] {Pan}        => {Agua}  0.6666667 0.8        1.20 4
[3] {Leche}      => {Pan}   0.6666667 0.8        0.96 4
[4] {Pan}        => {Leche} 0.6666667 0.8        0.96 4
[5] {Agua,Leche} => {Pan}   0.5000000 1.0        1.20 3
```

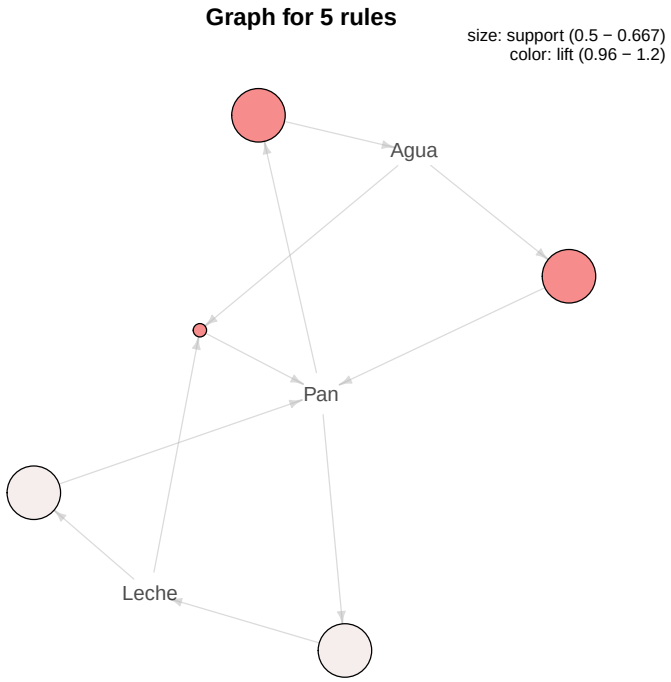
```
> plot(shop_data, method = "two-key plot")
```




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




```
> plot(shop_data, method = "graph")
```



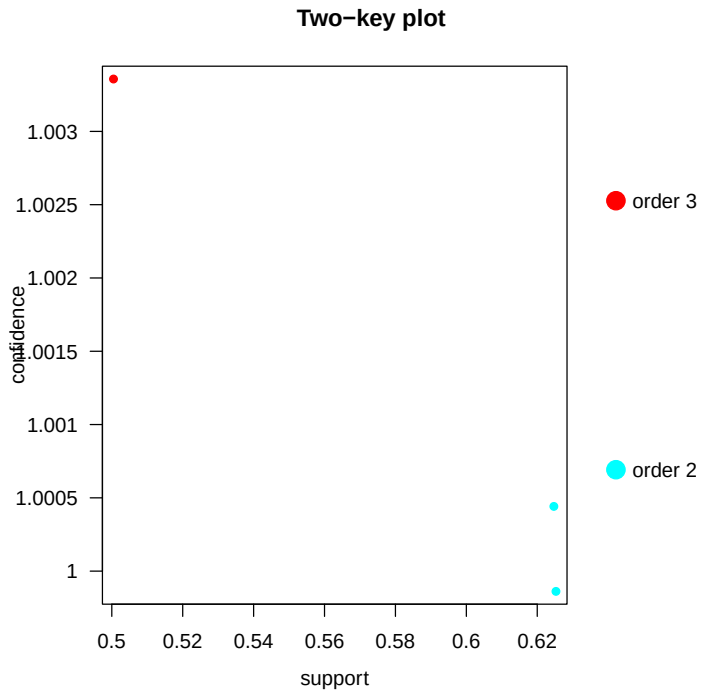
2 Asociación sobre los datos de los vehículos

Ya habíamos creado las función para leer datos de un .txt y suministrárselos a apriori. Repetimos el proceso ahora con los datos de los automóbiles iobteniendo los siguentes resultados para un soporte de 0.4 y una confianza de 0.9.

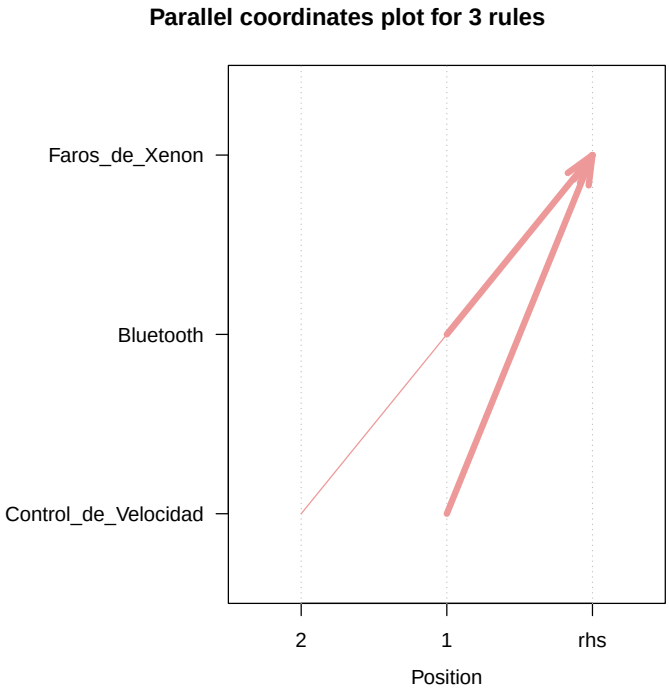
```
> car_data <<- calapriori(readAprioriFile("datos2.txt"),0.4,0.9)
```

```
    lhs                                  rhs              support confidence lift     count
[1] {Control_de_Velocidad}           => {Faros_de_Xenon} 0.625   1          1.333333 5
[2] {Bluetooth}                      => {Faros_de_Xenon} 0.625   1          1.333333 5
[3] {Bluetooth,Control_de_Velocidad} => {Faros_de_Xenon} 0.500   1          1.333333 4
```

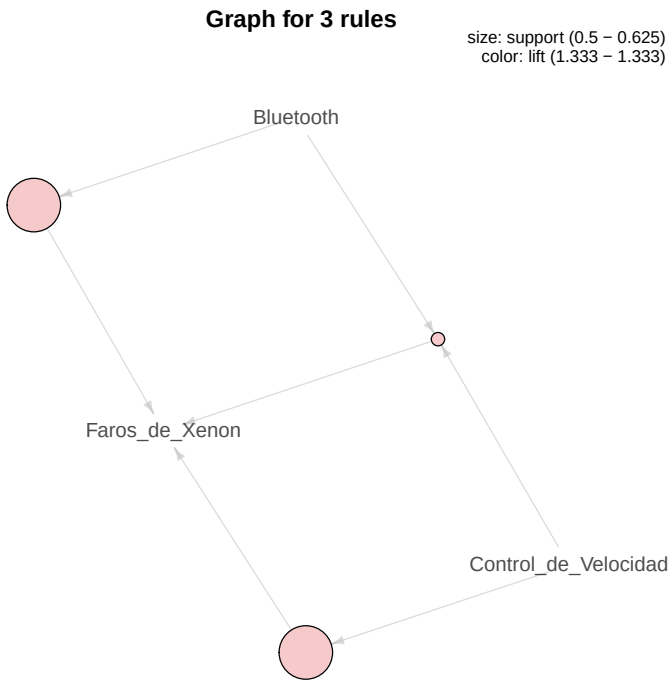
```
> plot(car_data, method = "two-key plot")
```




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




```
> plot(car_data, method = "graph")
```



3 Qué es apriori y cómo utilizarlo

El algoritmo a priori surge como respuesta al problema existente a la hora de analizar las reglas de asociación entre diferentes datos, puesto que con unas decenas de datos diferentes podemos encontrar miles de reglas diferentes y el principio en el que se basa es que si un conjunto de datos es frecuente, cualquier subconjunto del mismo también lo será. Este algoritmo encuentra reglas de asociación entre diferentes datos en base a dos parámetros que establecemos arbitrariamente que tienen que cumplirse en ellas, que son el soporte y la confianza. El soporte de un conjunto de datos se define como la proporción de apariciones que contiene dicho conjunto de datos en el espacio muestral. La confianza define la probabilidad de encontrar relacionadas una parte derecha de una regla de asociación con una parte izquierda determinadas.

Invocamos la función escribiendo apriori() y los diferentes parámetros que nos interesan para su uso son los siguientes: Apriori(data, parameter = NULL, appearance = NULL, control = NULL) Data: estructura de datos (por ejemplo, una matriz binaria o un data.frame) que se puede convertir en transacciones. Parameter: diferentes parámetros de la clase APparameter. El comportamiento por defecto de los parámetros importantes es: soporte (support) de 10Appearance: diferentes parámetros de la clase APappearance. Con esto podemos restringir la aparición de algunos datos. Control: diferentes parámetros de la clase APcontrol. Controla el rendimiento del algoritmo (sort, heap, filter...) y aspectos como la ordenación o el reporte del rendimiento (verbose).

4 Creación de un algoritmo apriori

Hemos programado una versión simplificada del algoritmo eliminando algunas de las optimizaciones que este realiza. La funcionalidad de nuestro algoritmo también es reducida y solo está pensado para tomar como entrada tablas de datos booleanos a diferencia del apriori original que puede tomar otros tipos de dato como entrada. Para implementarlo lo hemos hecho utilizando tres funciones.

Repetimos el cálculo de la asociación para los datos de las cestas de la compra y de los automóbiles comprobando que nuestro algoritmo proporciona los mismos resultados que apriori nos había dado anteriormente. Comprobamos que el algoritmo se comporta como esperábamos.

```
> print(toTable(f_apriori(readAprioriFile("datos1.txt"),0.5,0.8)),right=F)

  lhs             rhs     support   confidence lift count
1 {Pan}        => {Agua}  0.6666667 0.8        1.20 4
2 {Agua}       => {Pan}   0.6666667 1.0        1.20 4
3 {Pan}        => {Leche} 0.6666667 0.8        0.96 4
4 {Leche}      => {Pan}   0.6666667 0.8        0.96 4
5 {Agua,Leche} => {Pan}   0.5000000 1.0        1.20 3

> print(toTable(f_apriori(readAprioriFile("datos2.txt"),0.4,0.9)),right=F)

  lhs                                rhs              support confidence lift     count
1 {Bluetooth}                     => {Faros_de_Xenon} 0.625   1          1.333333 5
```

```
2 {Control_de_Velocidad}           => {Faros_de_Xenon} 0.625   1          1.333333 5
3 {Bluetooth,Control_de_Velocidad} => {Faros_de_Xenon} 0.500   1          1.333333 4
```

5 Apriori sobre un dataset real

Buscando informcación sobre el algoritmo apriori y sobre otros algoritmos de asociación hemos encontrado distintas fuentes de las que obtener datos típicos sobre los que aplicar algoritmos de asociación. Hemos elegido un conjunto de datos que están ya incluidos en el paquete arules para probar sobre ellos el comportamiento del algoritmo.

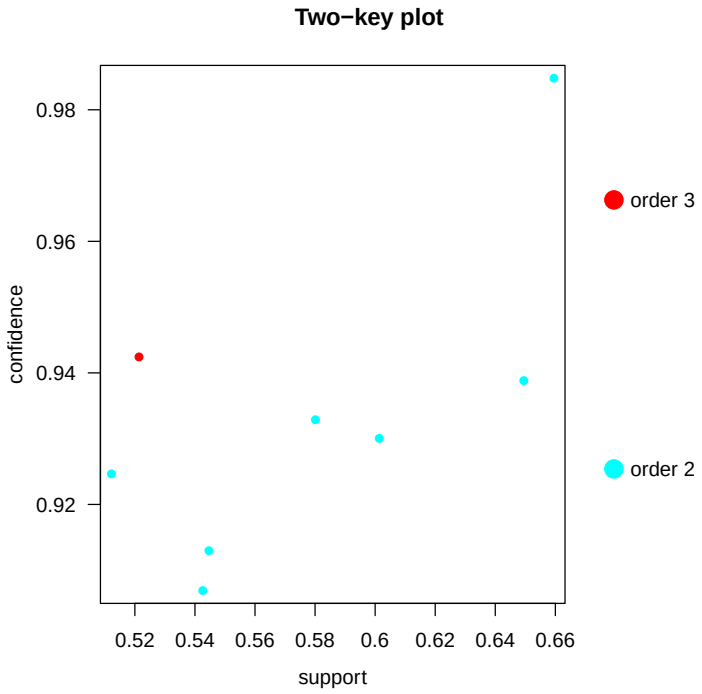
Podemos observar que la asociación puede aplicarse también sobre variables con múltiples valores, no solo verdadero o falso. Los datos contienen 8993 observaciones en 14 variables. El primer paso para procesarlos es crear clases de equivalencia (2) para cada valor. Una vez hecho esto solo nos queda crear las transacciones y proporcionárselas al algoritmo apriori.

```
> data("IncomeESL")
```

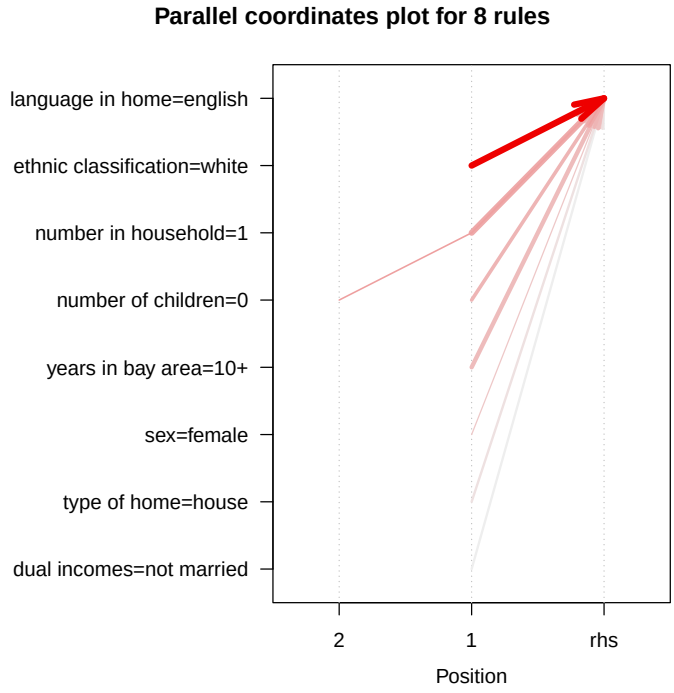
```
> income <- calapriori(IncomeESL, 0.5,0.9)
```

```
    lhs                              rhs                            support confidence     l
[1] {sex=female}                  => {language in home=english} 0.5122164  0.9246521 1.012
[2] {type of home=house}          => {language in home=english} 0.5446481  0.9129693 1.000
[3] {dual incomes=not married}    => {language in home=english} 0.5426120  0.9069033 0.993
[4] {number of children=0}        => {language in home=english} 0.5801338  0.9328812 1.021
[5] {years in bay area=10+}       => {language in home=english} 0.6013671  0.9300495 1.018
[6] {ethnic classification=white} => {language in home=english} 0.6595404  0.9847991 1.078
[7] {number in household=1}       => {language in home=english} 0.6495055  0.9388270 1.028
[8] {number in household=1,
     number of children=0}        => {language in home=english} 0.5213787  0.9424290 1.032
```

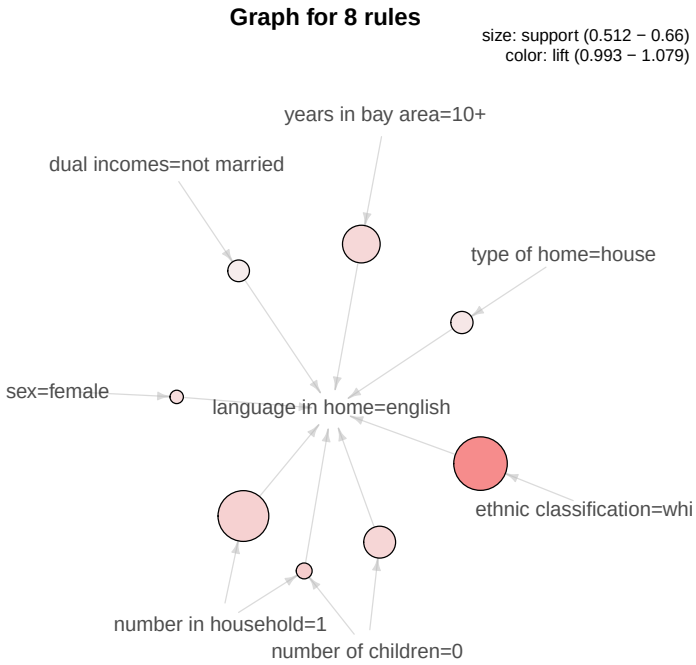
```
> plot(income, method = "two-key plot")
```




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




```
> plot(income, method = "graph")
```



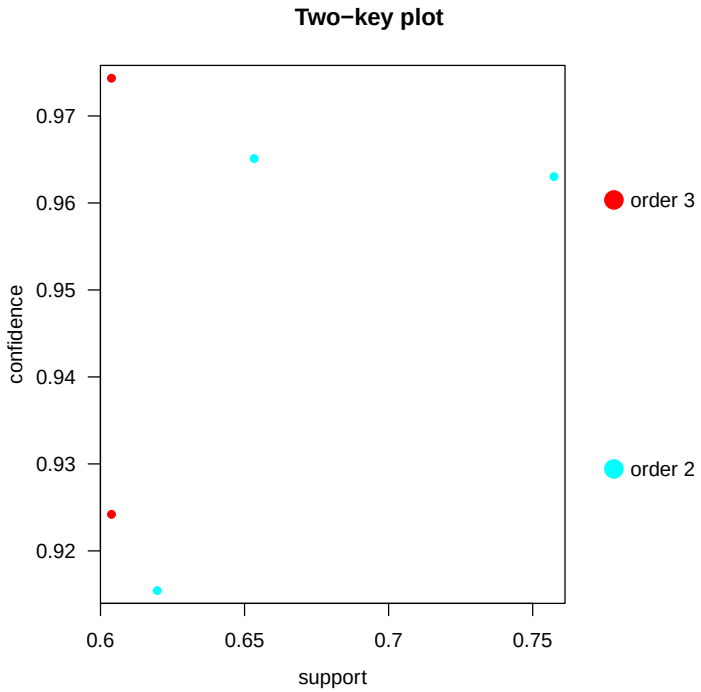
Haremos ahora el mismo proceso sobre datos de los supervivientes del titanic. En este caso, los datos encontrados ya están en formato de transacciones tay y como apriori los espera.

```
> load("./titanic.raw.rdata")
> titanic_data <- apriori(titanic.raw,
+       parameter=list(minlen=2, support=0.5,confidence=0.8),
+       control=list(verbose=F))
> inspect(titanic_data)
```

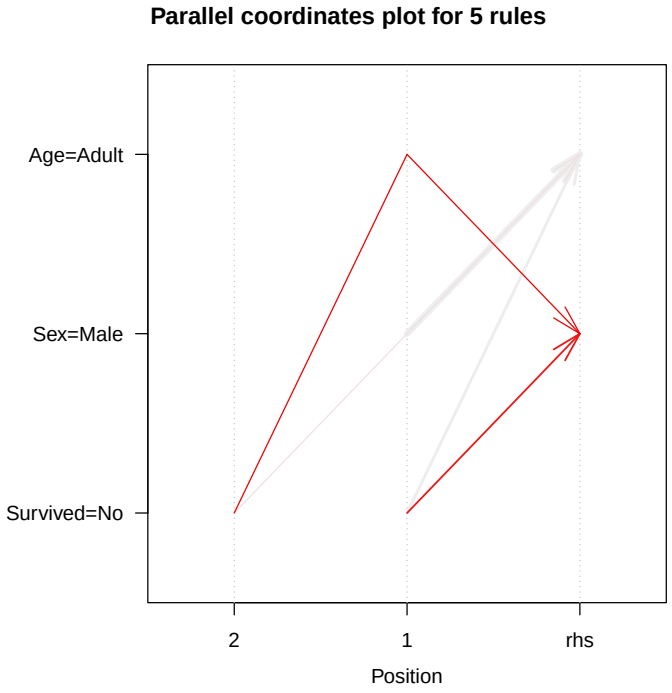
```
    lhs                          rhs         support   confidence lift     count
[1] {Survived=No}             => {Sex=Male}  0.6197183 0.9154362  1.163995 1364
[2] {Survived=No}             => {Age=Adult} 0.6533394 0.9651007  1.015386 1438
[3] {Sex=Male}                => {Age=Adult} 0.7573830 0.9630272  1.013204 1667
[4] {Sex=Male,Survived=No}    => {Age=Adult} 0.6038164 0.9743402  1.025106 1329
[5] {Age=Adult,Survived=No}   => {Sex=Male}  0.6038164 0.9242003  1.175139 1329
```

Podemos ver que los primeros en salvarse fueron las mujeres y los niños.

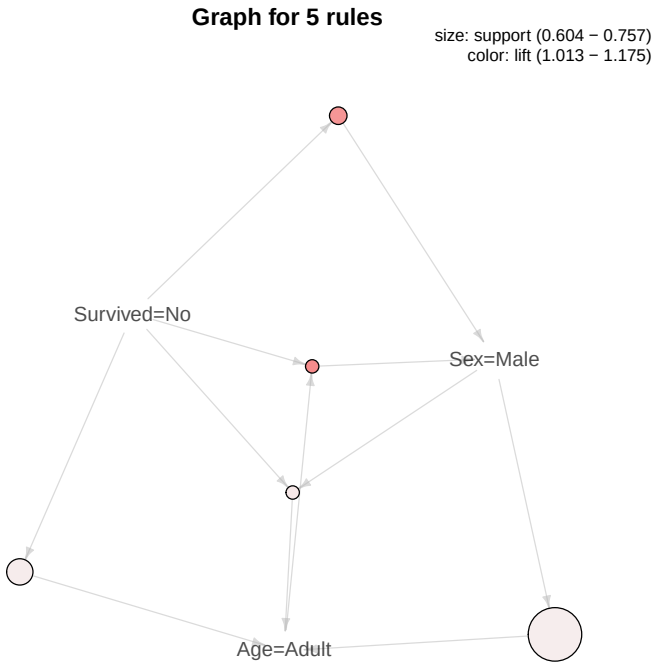
```
> plot(titanic_data, method = "two-key plot")
```




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




```
> plot(titanic_data, method = "graph")
```



6 Otros algoritmos de asociación

6.1 Que es eclat y cómo utilizarlo

Eclat es una alternativa a apriori que utiliza para cada elemento una lista en la que se registra en qué transacción está el elemento mencionado, reduciendo así enormemente el tiempo de cómputo, pero aumentando el consumo de memoria. La idea básica es que usa estas listas y realiza intersecciones con otras para calcular el soporte del elemento, evitando generar subconjuntos que no existan en el espacio muestral.

Invocamos la función escribiendo eclat() y los diferentes parámetros que nos interesan para su uso son los siguientes: Eclat(data, parameter = NULL, control = NULL) Data: estructura de datos (por ejemplo, una matriz binaria o un data.frame) que se puede convertir en transacciones. Parameter: diferentes parámetros de la clase ECparameter. El comportamiento por defecto de los parámetros importantes es: soporte (support) de 10Control: diferentes parámetros de la clase ECcontrol. Controla el rendimiento del algoritmo (sort, heap, filter...) y aspectos como la ordenación o el reporte del rendimiento (verbose).

6.1.1 Ejemplos de con eclat

Ejecución sobre la muestra de los datos de las cestas de la compra.

```
> caleclat(readAprioriFile("datos1.txt"),0.5)

    items             support   count
[1] {Pan,Agua,Leche} 0.5000000 3
```

```
[2] {Pan,Agua}       0.6666667 4
[3] {Agua,Leche}     0.5000000 3
[4] {Pan,Leche}      0.6666667 4
set of 4 itemsets
```

Ejecución sobre la muestra de los datos de los componentes de los coches.

```
> caleclat(readAprioriFile("datos2.txt"),0.4)
```

```
    items                                                 support count
[1] {Faros_de_Xenon,Bluetooth,Control_de_Velocidad} 0.500   4
[2] {Faros_de_Xenon,Control_de_Velocidad}           0.625   5
[3] {Bluetooth,Control_de_Velocidad}                0.500   4
[4] {Faros_de_Xenon,Bluetooth}                      0.625   5
set of 4 itemsets
```

Ejecución sobre la muestra de los datos demográficos incluídos en el paquete arules.

```
> caleclat(IncomeESL, 0.5)
```

```
     items                              support count
[1]  {sex=female,
      language in home=english}      0.5122164  3522
[2]  {age=14-34,
      language in home=english}      0.5248691  3609
[3]  {type of home=house,
      language in home=english}      0.5446481  3745
[4]  {dual incomes=not married,
      language in home=english}      0.5426120  3731
[5]  {number in household=1,
      number of children=0,
      language in home=english}      0.5213787  3585
[6]  {number of children=0,
      language in home=english}      0.5801338  3989
[7]  {number in household=1,
      number of children=0}          0.5532286  3804
[8]  {income=$0-$40,000,
      language in home=english}      0.5578825  3836
[9]  {income=$0-$40,000,
      education=no college graduate} 0.5018906  3451
[10] {years in bay area=10+,
      language in home=english}      0.6013671  4135
[11] {ethnic classification=white,
      language in home=english}      0.6595404  4535
[12] {number in household=1,
      ethnic classification=white}   0.5001454  3439
[13] {number in household=1,
      language in home=english}      0.6495055  4466
[14] {education=no college graduate,
      language in home=english}      0.6343805  4362
set of 14 itemsets
```

Ejecución sobre la muestra de los datos de los supervivientes del titanic.

```
> inspect(eclat(titanic.raw, parameter=list(minlen=2, support=0.5),
+                            control=list(verbose=F)))

    items                               support   count
[1] {Sex=Male,Age=Adult,Survived=No} 0.6038164 1329
[2] {Age=Adult,Survived=No}          0.6533394 1438
[3] {Sex=Male,Survived=No}           0.6197183 1364
[4] {Sex=Male,Age=Adult}             0.7573830 1667
```

Podemos observar que obtenemos el mismo soporte que con apriori y que tal y como hemos dicho la confianza no es calculada. Para compara los resultados de apriori con los de eclat debemos buscar en apriori los conjuntos que se correspondan con los de eclar.

Ej: los conjuntos de apriori: {Pan} => {Agua} y {Agua} => {Pan} en eclat serán solo uno: Pan, Agua