Ejemplo de reglas de asociación

Contents

En este cuaderno mostraremos un ejemplo sencillo de extracción de reglas de asociación con el conjunto de datos de las cestas de la compra, utilizado en las diapositivas de la asignatura.

Nos basaremos en las funcionalidades proporcionadas por el paquete `arules()`.

En primer lugar, asignamos una constante para el nombre del fichero:

```
fichero <- "cesta.csv"
```

Lectura de datos

Los datos del fichero cesta.csv se presentan en forma de transacciones, como se puede ver:

```
readLines(fichero) %>% paste0(collapse="\n") %>% cat
```

```
## Transacciones
## pan,leche,pañales
## pan,pañales,cerveza,huevos
## leche,pañales,cerveza,refresco,café
## pan,leche,pañales,cerveza
## pan,refresco,leche,pañales
```

`arules()` ofrece la función `read.transactions()` para lectura de ficheros con transacciones:

```
datost <- read.transactions(fichero, sep=",", header = TRUE)
```

Podemos obtener un resumen de las transacciones con `summary()`:

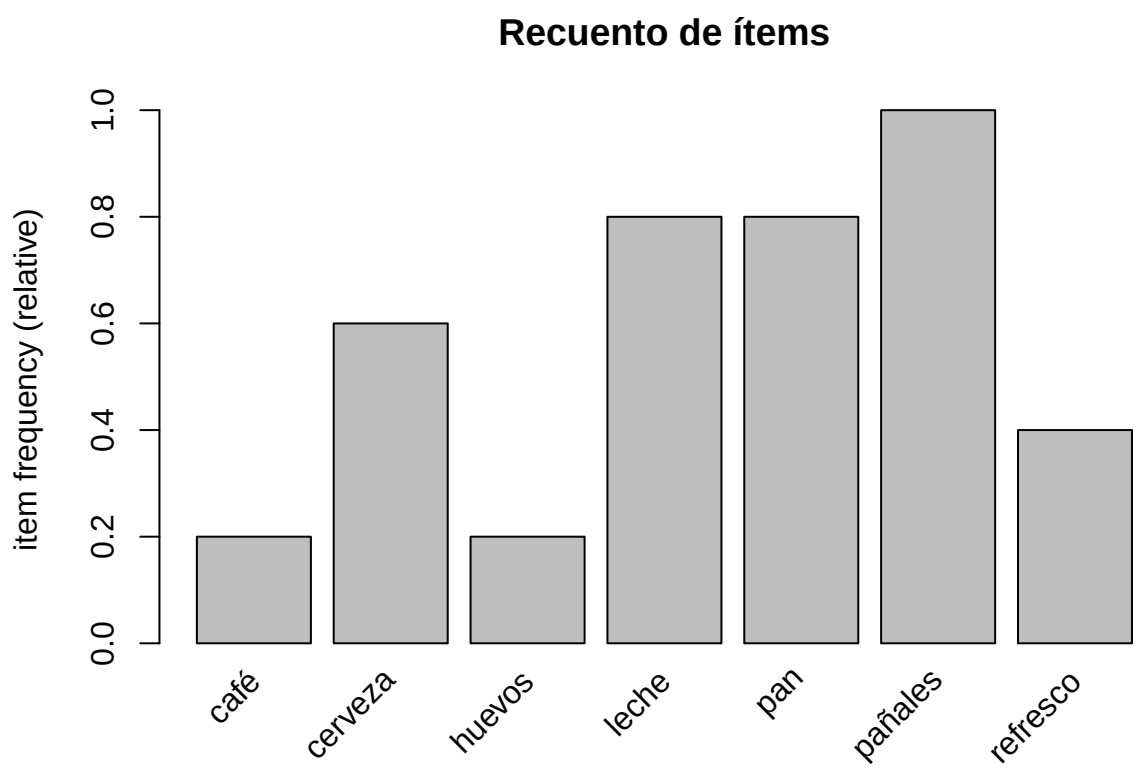
```
summary(datost)
```

```
## transactions as itemMatrix in sparse format with
##  5 rows (elements/itemsets/transactions) and
```

```
##  7 columns (items) and a density of 0.5714286
##
## most frequent items:
##  pañales    leche      pan  cerveza refresco  (Other)
##        5        4        4        3        2        2
##
## element (itemset/transaction) length distribution:
## sizes
## 3 4 5
## 1 3 1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       3       4       4       4       4       5
##
## includes extended item information - examples:
##    labels
## 1    café
## 2 cerveza
## 3  huevos
```

Gráficamente, podemos ver los ítems frecuentes de las transacciones:

```
itemFrequencyPlot(datost, type="relative", main="Recuento de ítems")
```



Listado de transacciones

Para ver y operar con las las transacciones, convertimos `datost` a una lista con `LIST()`:

```
listat <- LIST(datost)
listat
```

```
## [[1]]
## [1] "leche"   "pan"     "pañales"
##
## [[2]]
## [1] "cerveza" "huevos"  "pan"     "pañales"
##
## [[3]]
## [1] "café"     "cerveza"  "leche"    "pañales"  "refresco"
##
## [[4]]
## [1] "cerveza" "leche"   "pan"     "pañales"
##
## [[5]]
## [1] "leche"    "pan"      "pañales"  "refresco"
```

Si solo queremos trabajar con un subconjunto de transacciones, podemos hacer una sublista:

```
print("Transacción número 5: ")
```

```
## [1] "Transacción número 5: "
```

```
listat[[5]]
```

```
## [1] "leche"    "pan"      "pañales"  "refresco"
```

```
print("Transacciones 1, 3, 5")
```

```
## [1] "Transacciones 1, 3, 5"
```

```
listat[c(1, 3, 5)]
```

```
## [[1]]
## [1] "leche"   "pan"     "pañales"
##
## [[2]]
## [1] "café"     "cerveza"  "leche"    "pañales"  "refresco"
##
## [[3]]
## [1] "leche"    "pan"      "pañales"  "refresco"
```

Y seleccionar basándonos en criterios más complejos con `which()` y `grepl()`:

```
con_pan <- which(grepl("pan", listat))
print("Transacciones que incluyen pan")
```

```
## [1] "Transacciones que incluyen pan"
```

```
listat[con_pan]
```

```
## [[1]]
## [1] "leche"   "pan"     "pañales"
##
## [[2]]
## [1] "cerveza" "huevos"  "pan"     "pañales"
##
## [[3]]
## [1] "cerveza" "leche"   "pan"     "pañales"
##
## [[4]]
```

```
## [1] "leche"    "pan"      "pañales"  "refresco"
```

```
con_pan_y_leche <- which(grepl("pan", listat) & grepl("leche", listat))
print("Transacciones que incluyen pan y leche")
```

```
## [1] "Transacciones que incluyen pan y leche"
```

```
listat[con_pan_y_leche]
```

```
## [[1]]
## [1] "leche"   "pan"     "pañales"
##
## [[2]]
## [1] "cerveza" "leche"   "pan"     "pañales"
##
## [[3]]
## [1] "leche"    "pan"      "pañales"  "refresco"
```

Itemsets frecuentes

Para obtener los itemsets frecuentes, usamos la función `apriori()` con el `target="frequent itemsets"`:

```
minSup <- 3/length(listat)

itemsetsFrecuentes <- apriori(datost, parameter = list(support = minSup, target = "frequent itemsets"))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##          NA    0.1    1 none FALSE            TRUE       5     0.6      1
##  maxlen            target  ext
##      10 frequent itemsets TRUE
##
## 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 ...[7 item(s), 5 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].
## sorting transactions ... done [0.00s].
## writing ... [9 set(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
inspect(itemsetsFrecuentes)[c(1, 2, 4)]
```

```
##     items               support transIdenticalToItemsets count
## [1] {cerveza}           0.6     0.0                       3
## [2] {pan}               0.8     0.0                       4
## [3] {leche}             0.8     0.0                       4
## [4] {pañales}           1.0     0.0                       5
```

```
## [5] {cerveza,pañales}   0.6     0.0                     3
## [6] {leche,pan}         0.6     0.0                     3
## [7] {pan,pañales}       0.8     0.0                     4
## [8] {leche,pañales}     0.8     0.0                     4
## [9] {leche,pan,pañales} 0.6     0.4                     3
```

```
##                   items support count
## [1]           {cerveza}     0.6     3
## [2]               {pan}     0.8     4
## [3]             {leche}     0.8     4
## [4]           {pañales}     1.0     5
## [5]   {cerveza,pañales}     0.6     3
## [6]         {leche,pan}     0.6     3
## [7]       {pan,pañales}     0.8     4
## [8]     {leche,pañales}     0.8     4
## [9] {leche,pan,pañales}     0.6     3
```

Extracción de reglas

Extracción general

Para obtener las reglas de asociación, lanzamos `apriori()` cambiando el `target` a `rules`:

```
minSup  <- 3/length(listat)
minConf <- 0.75

reglas <- apriori(datost,
                  parameter = list(support = minSup,
                                   confidence = minConf,
                                   target = "rules",
                                   minlen = 2,
                                   maxlen = 5))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.75    0.1    1 none FALSE            TRUE       5     0.6      2
##  maxlen target  ext
##       5  rules TRUE
##
## 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 ...[7 item(s), 5 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 ... [10 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
summary(reglas)
```

```
## set of 10 rules
##
## rule length distribution (lhs + rhs):sizes
## 2 3
## 7 3
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00    2.00    2.00    2.30    2.75    3.00
##
## summary of quality measures:
##     support       confidence      coverage        lift            count
##  Min.   :0.60   Min.   :0.75   Min.   :0.6   Min.   :0.9375   Min.   :3.0
##  1st Qu.:0.60   1st Qu.:0.75   1st Qu.:0.8   1st Qu.:0.9375   1st Qu.:3.0
##  Median :0.60   Median :0.80   Median :0.8   Median :1.0000   Median :3.0
##  Mean   :0.68   Mean   :0.86   Mean   :0.8   Mean   :0.9750   Mean   :3.4
##  3rd Qu.:0.80   3rd Qu.:1.00   3rd Qu.:0.8   3rd Qu.:1.0000   3rd Qu.:4.0
##  Max.   :0.80   Max.   :1.00   Max.   :1.0   Max.   :1.0000   Max.   :4.0
##
## mining info:
##    data ntransactions support confidence
##  datost             5     0.6       0.75
```

Visualización de reglas

Formato de tabla

Para mostrar las reglas en forma de tabla usamos `inspect()`:

```
inspect(reglas)
```

```
##      lhs                rhs       support confidence coverage lift   count
## [1]  {cerveza}       => {pañales} 0.6     1.00       0.6      1.0000 3
## [2]  {pan}           => {leche}   0.6     0.75       0.8      0.9375 3
## [3]  {leche}         => {pan}     0.6     0.75       0.8      0.9375 3
## [4]  {pan}           => {pañales} 0.8     1.00       0.8      1.0000 4
## [5]  {pañales}       => {pan}     0.8     0.80       1.0      1.0000 4
## [6]  {leche}         => {pañales} 0.8     1.00       0.8      1.0000 4
## [7]  {pañales}       => {leche}   0.8     0.80       1.0      1.0000 4
## [8]  {leche,pan}     => {pañales} 0.6     1.00       0.6      1.0000 3
## [9]  {pan,pañales}   => {leche}   0.6     0.75       0.8      0.9375 3
## [10] {leche,pañales} => {pan}     0.6     0.75       0.8      0.9375 3
```

El paquete `dplyr` no se lleva bien con las columnas sin nombre, por lo que lo primero que haremos es renombrar la columna 2 (la implicación):

```
reglas_df <- inspect(reglas) %>%
  rename("implies" = 2)
```

```
##      lhs                rhs       support confidence coverage lift   count
## [1]  {cerveza}       => {pañales} 0.6     1.00       0.6      1.0000 3
## [2]  {pan}           => {leche}   0.6     0.75       0.8      0.9375 3
## [3]  {leche}         => {pan}     0.6     0.75       0.8      0.9375 3
## [4]  {pan}           => {pañales} 0.8     1.00       0.8      1.0000 4
```

```
## [5]  {pañales}       => {pan}     0.8     0.80       1.0      1.0000 4
## [6]  {leche}         => {pañales} 0.8     1.00       0.8      1.0000 4
## [7]  {pañales}       => {leche}   0.8     0.80       1.0      1.0000 4
## [8]  {leche,pan}     => {pañales} 0.6     1.00       0.6      1.0000 3
## [9]  {pan,pañales}   => {leche}   0.6     0.75       0.8      0.9375 3
## [10] {leche,pañales} => {pan}     0.6     0.75       0.8      0.9375 3
```

Ahora ya podemos manipular la tabla con instrucciones `filter()` y búsqueda de texto con `grepl()`:

```
# reglas con alto soporte
reglas_frecuentes <- reglas_df %>%
  filter(support > 0.6)
reglas_frecuentes
```

```
##           lhs implies       rhs support confidence coverage lift count
## [4]     {pan}      => {pañales}     0.8        1.0      0.8    1     4
## [5] {pañales}      =>     {pan}     0.8        0.8      1.0    1     4
## [6]   {leche}      => {pañales}     0.8        1.0      0.8    1     4
## [7] {pañales}      =>   {leche}     0.8        0.8      1.0    1     4
```

```
# reglas con alta confianza
reglas_confianza <- reglas_df %>%
  filter(confidence > 0.8)
reglas_confianza
```

```
##             lhs implies       rhs support confidence coverage lift count
## [1]   {cerveza}      => {pañales}     0.6          1      0.6    1     3
## [4]       {pan}      => {pañales}     0.8          1      0.8    1     4
## [6]     {leche}      => {pañales}     0.8          1      0.8    1     4
## [8] {leche,pan}      => {pañales}     0.6          1      0.6    1     3
```

```
# reglas con un item determinado en el consecuente
reglas_pan <- reglas_df %>%
  filter(grepl("pan", rhs))
reglas_pan
```

```
##                  lhs implies   rhs support confidence coverage   lift count
## [3]          {leche}      => {pan}     0.6       0.75      0.8 0.9375     3
## [5]        {pañales}      => {pan}     0.8       0.80      1.0 1.0000     4
## [10] {leche,pañales}      => {pan}     0.6       0.75      0.8 0.9375     3
```

```
# reglas con algún item determinado en el antecedente
reglas_panOleche <- reglas_df %>%
  filter(grepl("pan", lhs) | grepl("leche", lhs))
reglas_panOleche
```

```
##                  lhs implies       rhs support confidence coverage   lift count
## [2]            {pan}      =>   {leche}     0.6       0.75      0.8 0.9375     3
## [3]          {leche}      =>     {pan}     0.6       0.75      0.8 0.9375     3
## [4]            {pan}      => {pañales}     0.8       1.00      0.8 1.0000     4
## [6]          {leche}      => {pañales}     0.8       1.00      0.8 1.0000     4
## [8]      {leche,pan}      => {pañales}     0.6       1.00      0.6 1.0000     3
## [9]    {pan,pañales}      =>   {leche}     0.6       0.75      0.8 0.9375     3
## [10] {leche,pañales}      =>     {pan}     0.6       0.75      0.8 0.9375     3
```

```
# reglas con un itemset determinado en el antecedente
reglas_panYleche <- reglas_df %>%
  filter(grepl("pan", lhs) & grepl("leche", lhs))
```

```
reglas_panYleche
```

```
##             lhs implies       rhs support confidence coverage lift count
## [8] {leche,pan}      => {pañales}     0.6          1      0.6    1     3
```

También podemos ordenar la tabla de reglas con `arrange()`:

```
reglas_ordenadas_confianza <- reglas_df %>%
  arrange(desc(confidence))
reglas_ordenadas_confianza
```

```
##                  lhs implies       rhs support confidence coverage   lift count
## [1]        {cerveza}      => {pañales}     0.6       1.00      0.6 1.0000     3
## [4]            {pan}      => {pañales}     0.8       1.00      0.8 1.0000     4
## [6]          {leche}      => {pañales}     0.8       1.00      0.8 1.0000     4
## [8]      {leche,pan}      => {pañales}     0.6       1.00      0.6 1.0000     3
## [5]        {pañales}      =>     {pan}     0.8       0.80      1.0 1.0000     4
## [7]        {pañales}      =>   {leche}     0.8       0.80      1.0 1.0000     4
## [2]            {pan}      =>   {leche}     0.6       0.75      0.8 0.9375     3
## [3]          {leche}      =>     {pan}     0.6       0.75      0.8 0.9375     3
## [9]    {pan,pañales}      =>   {leche}     0.6       0.75      0.8 0.9375     3
## [10] {leche,pañales}      =>     {pan}     0.6       0.75      0.8 0.9375     3
```

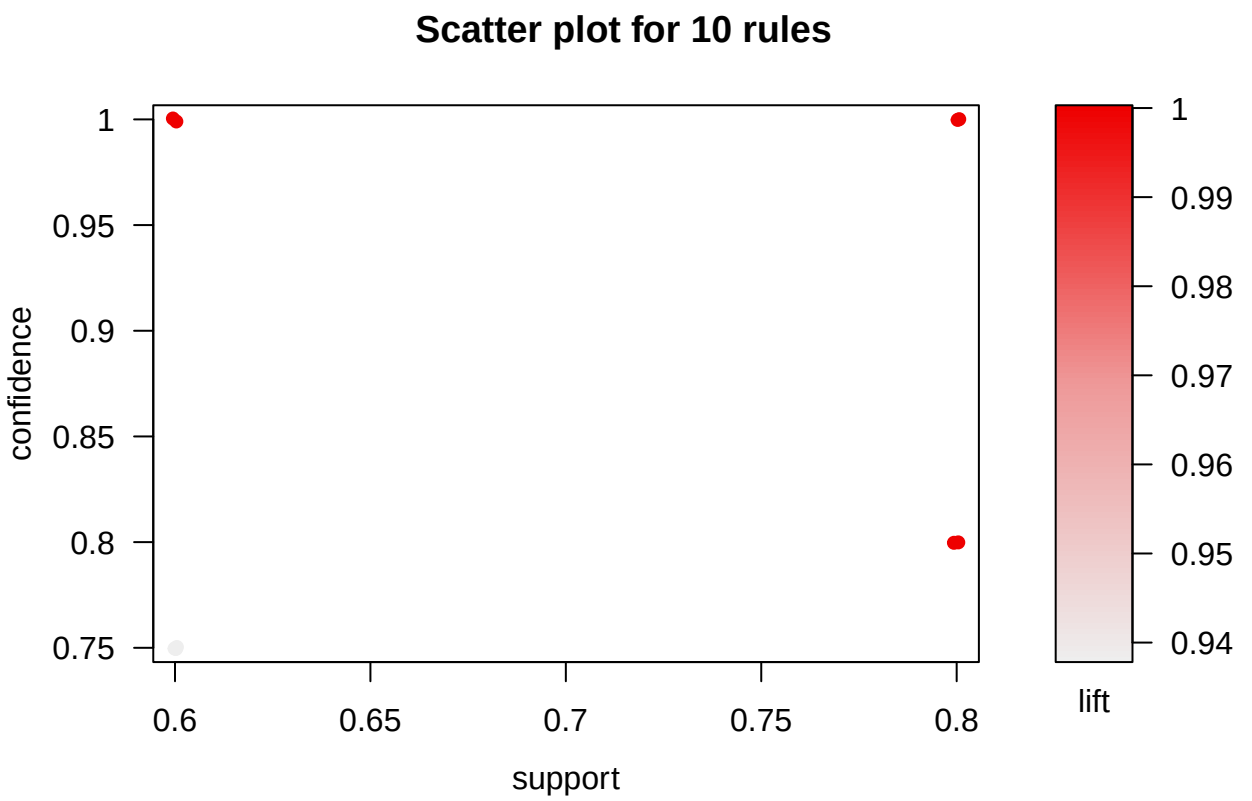
Formato visual

El paquete `arulesViz` incluye varias funcionalidades para la visualización de reglas de asociación.

La más directa es `plot()`, que muestra las reglas obtenidas sobre un diagrama de dispersión con x=soporte, y=confianza.

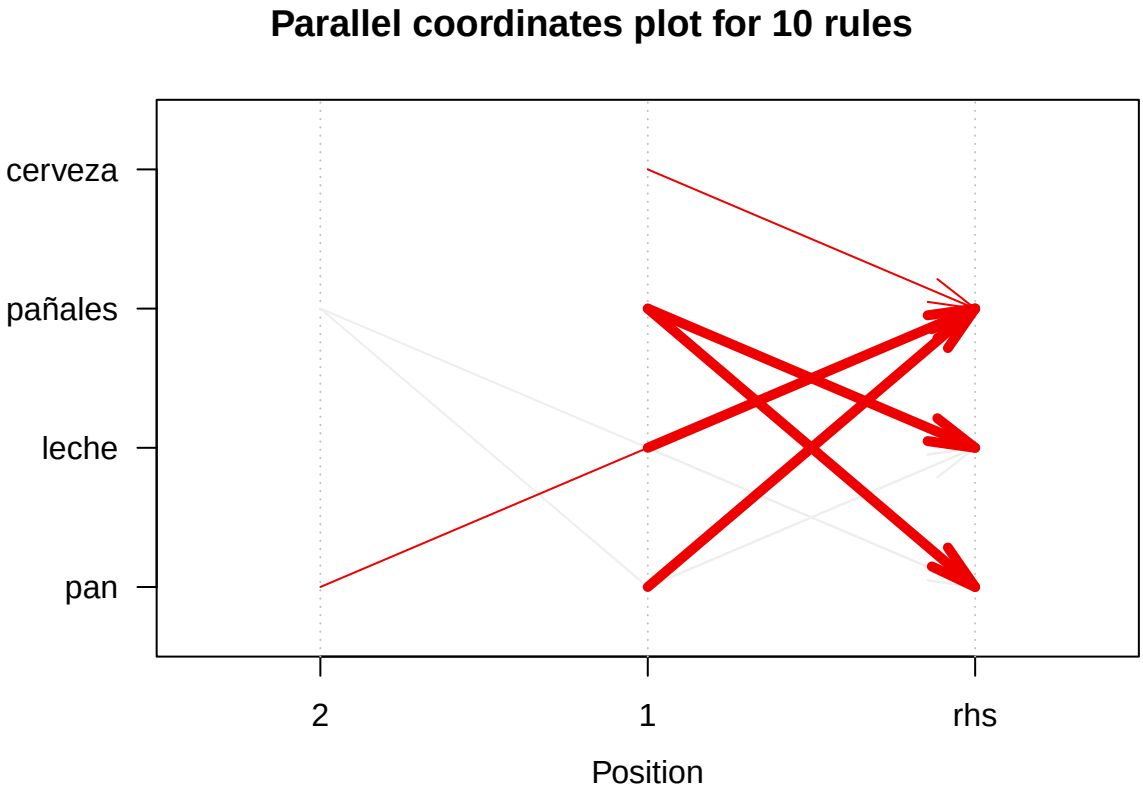
```
library(arulesViz)   # install.packages("arulesViz")
plot(reglas)
```

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```



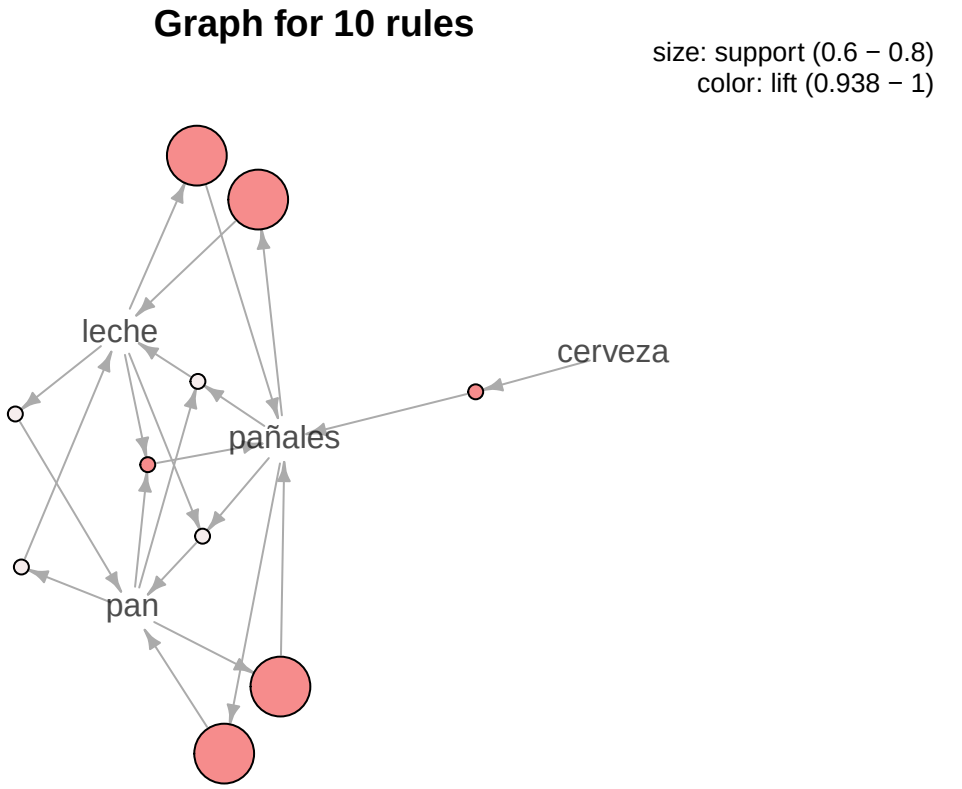
`plot()` ofrece varios parámetros para modificar la visualización y el tipo de gráfico generado. Por ejemplo, para formato de coordenadas paralelas:

```
plot(reglas, method="paracoord")     # method="two-key plot"
```



Podemos ver las reglas en formato de grafo. Los nodos del grafo representan reglas (círculos) e ítems (etiquetas de texto); los arcos del grafo representan participación en la regla (antecedente o consecuente, según la dirección del arco).

```
plot(reglas, method="graph")
```



Estos gráficos se pueden hacer interactivos añadiendo el motor HTML para renderizado con `engine = "htmlwidget"`:

```
plot(reglas, method="graph", engine = "htmlwidget")
```

Los grafos de reglas pueden exportarse con `saveAsGraph()` a formato *graphml* , que puede ser leído desde Gephi:

```
saveAsGraph(reglas, file = "reglas.graphml")
```

Extensiones del algoritmo de generación

Extracción centrada en ciertos ítems

Podemos limitar la ejecución del algoritmo para que solo se tengan en cuenta ciertos ítems en el antecedente o el consecuente de las reglas mediante el parámetro `appearance`. En este ejemplo, generamos reglas con leche en el consecuente:

```
reglas_leche <- apriori(datost,
                        parameter = list(support=0.5, confidence=0.5, minlen=2),
                        appearance = list(default="lhs", rhs="leche"))
```

```
## Apriori
##
## Parameter specification:
```

```
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5     0.5      2
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 2
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[7 item(s), 5 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].
```

```
inspect(reglas_leche)
```

```
##     lhs              rhs     support confidence coverage lift   count
## [1] {pan}         => {leche} 0.6     0.75       0.8      0.9375 3
## [2] {pañales}     => {leche} 0.8     0.80       1.0      1.0000 4
## [3] {pan,pañales} => {leche} 0.6     0.75       0.8      0.9375 3
```

Los resultados son los mismos que si generamos todas las reglas y después filtramos por ítem, aunque el procedimiento es más rápido.

Eliminado de reglas más amplias que otras

Puede ocurrir que aparezcan reglas que están incluidas dentro de otras reglas. Para identificarlas, usamos la función `is.subset()`:

```
is.subset(reglas, reglas)
```

```
## 10 x 10 sparse Matrix of class "ngCMatrix"
##    [[ suppressing 10 column names '{cerveza,pañales}', '{leche,pan}', '{leche,pan}' ... ]]
##
## {cerveza,pañales}   | . . . . . . . . .
## {leche,pan}         . | | . . . . | | |
## {leche,pan}         . | | . . . . | | |
## {pan,pañales}       . . . | | . . | | |
## {pan,pañales}       . . . | | . . | | |
## {leche,pañales}     . . . . . | | | | |
## {leche,pañales}     . . . . . | | | | |
## {leche,pan,pañales} . . . . . . . | | |
## {leche,pan,pañales} . . . . . . . | | |
## {leche,pan,pañales} . . . . . . . | | |
```

Si queremos eliminarlas, podemos ver qué reglas tienen un recuento de "inclusión" mayor que 1:

```
ya_incluidas <- which(colSums(is.subset(reglas, reglas)) > 1)
ya_incluidas
```

```
##         {leche,pan}         {leche,pan}       {pan,pañales}       {pan,pañales}
```

```
##                   2                   3                   4                   5
##     {leche,pañales}     {leche,pañales} {leche,pan,pañales} {leche,pan,pañales}
##                   6                   7                   8                   9
## {leche,pan,pañales}
##                  10
```

Y, a continuación, eliminarlas:

```
reglas_no_incluidas <- reglas[-ya_incluidas]
inspect(reglas_no_incluidas)
```

```
##     lhs          rhs       support confidence coverage lift count
## [1] {cerveza} => {pañales} 0.6     1          0.6      1    3
```