LaboratorioReglasAsociacion

Pollo

20/1/2017

Reglas de Asociación

Cargamos la BD en nuestra zona de trabajo y consultamos sus dimensiones.Vemos las 2 primeras filas para ver los atributos y sus tipos

```
library(arules)
data()
data("AdultUCI")
```

```
dim(AdultUCI)
```

```
## [1] 48842    15
```

```
AdultUCI[1:2,]
```

```
##   age        workclass fnlwgt education education-num     marital-status
## 1  39        State-gov  77516 Bachelors            13      Never-married
## 2  50 Self-emp-not-inc  83311 Bachelors            13 Married-civ-spouse
##        occupation  relationship  race  sex capital-gain capital-loss
## 1    Adm-clerical Not-in-family White Male         2174            0
## 2 Exec-managerial       Husband White Male            0            0
##   hours-per-week native-country income
## 1             40  United-States  small
## 2             13  United-States  small
```

De los 6 atributos continuos: 2 los eliminamos porque aporan información redundante: fnlwgt y education-num y los 4 restantes los dividimos en intervalos. Finalmente convertimos el data.frame en un conjunto de transacciones con la función as

```
AdultUCI[["fnlwgt"]] = NULL
AdultUCI[["education-num"]] = NULL
AdultUCI[[ "age"]] = ordered( cut ( AdultUCI[[ "age"]], c(0,25,45,65,100) ) ,
        labels = c ("Young", "Middle-aged", "Senior", "Old"))
AdultUCI[[ "hours-per-week"]] = ordered( cut ( AdultUCI[[ "hours-per-week"]],
        c(0,25,40,60,168) ) ,
        labels = c("Part-time", "Full-time", "Over-time", "Workaholic"))
AdultUCI[[ "capital-gain"]] = ordered( cut ( AdultUCI[[ "capital-gain"]],
        c(-Inf,0,median(AdultUCI[[ "capital-gain"]][AdultUCI[[ "capital-gain"]]>0]), Inf) ) ,
        labels = c("None", "Low", "High"))
AdultUCI[[ "capital-loss"]] = ordered( cut ( AdultUCI[[ "capital-loss"]],
        c(-Inf,0, median(AdultUCI[[ "capital-loss"]][AdultUCI[[ "capital-loss"]]>0]), Inf) ) ,
        labels = c("None", "Low", "High"))
Adult <- as(AdultUCI,"transactions")
Adult
```

```
## transactions in sparse format with
##  48842 transactions (rows) and
##  115 items (columns)
```

Vemos el resumen de la BD

```r
summary(Adult)
```

```
## transactions as itemMatrix in sparse format with
##  48842 rows (elements/itemsets/transactions) and
##  115 columns (items) and a density of 0.1089939
##
## most frequent items:
##            capital-loss=None            capital-gain=None
##                        46560                        44807
## native-country=United-States                   race=White
##                        43832                        41762
##            workclass=Private                      (Other)
##                        33906                       401333
##
## element (itemset/transaction) length distribution:
## sizes
##     9    10    11    12    13
##    19   971  2067 15623 30162
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9.00   12.00   13.00   12.53   13.00   13.00
##
## includes extended item information - examples:
##            labels variables      levels
## 1       age=Young       age       Young
## 2 age=Middle-aged       age Middle-aged
## 3      age=Senior       age      Senior
##
## includes extended transaction information - examples:
##   transactionID
## 1             1
## 2             2
## 3             3
```

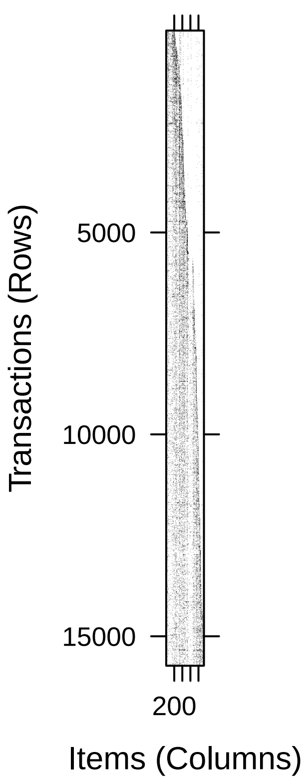
Representar gráficamente la distribución de los items en las transacciones. Como en Adult cada transacción tienen un valor cada atributo/variable, usamos para probarlo la BD Epub (15729 transacciones y 936 items)

```r
data(Epub)
summary(Epub)
```

```
## transactions as itemMatrix in sparse format with
##  15729 rows (elements/itemsets/transactions) and
##  936 columns (items) and a density of 0.001758755
##
## most frequent items:
## doc_11d doc_813 doc_4c6 doc_955 doc_698 (Other)
```

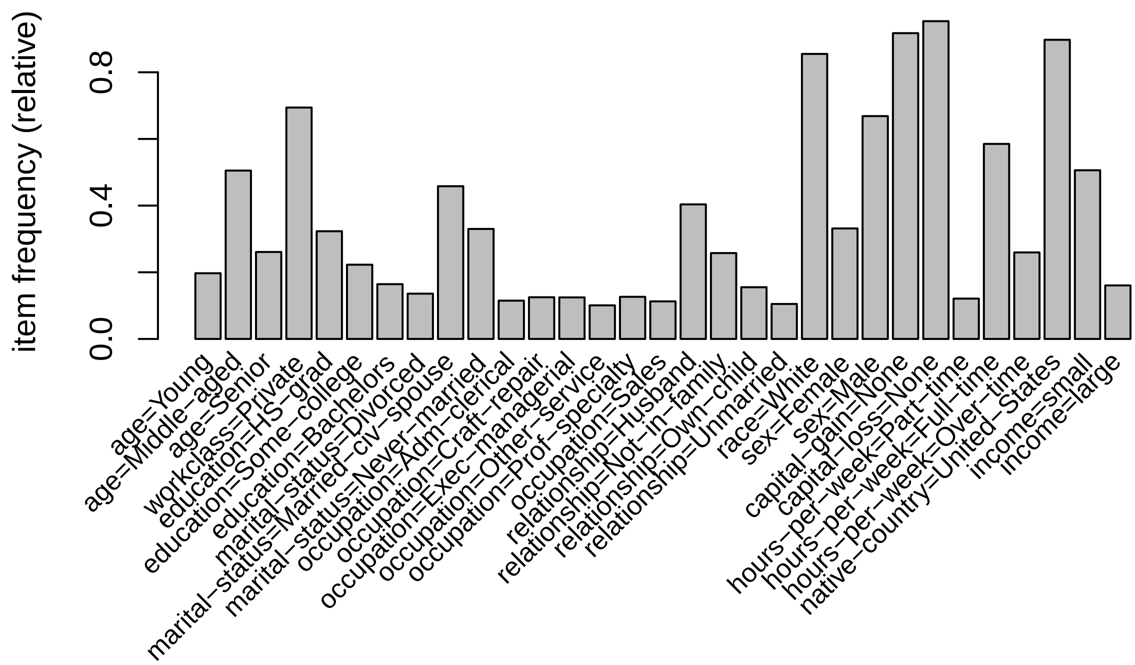
```
##      356      329      288      282      245    24393
##
## element (itemset/transaction) length distribution:
## sizes
##      1      2      3      4      5      6      7      8      9     10     11     12
## 11615   2189    854    409    198    121     93     50     42     34     26     12
##     13     14     15     16     17     18     19     20     21     22     23     24
##     10     10      6      8      6      5      8      2      2      3      2      3
##     25     26     27     28     30     34     36     38     41     43     52     58
##      4      5      1      1      1      2      1      2      1      1      1      1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.646   2.000  58.000
##
## includes extended item information - examples:
##    labels
## 1 doc_11d
## 2 doc_13d
## 3 doc_14c
##
## includes extended transaction information - examples:
##       transactionID           TimeStamp
## 10792  session_4795 2003-01-02 02:59:00
## 10793  session_4797 2003-01-02 13:46:01
## 10794  session_479a 2003-01-02 16:50:38
```

```r
image(Epub)
```



Para ver gráficamente que items son los más importantes: donde el mínimo soporte será 0.1 y reducimos el tamaño de los títulos

```
itemFrequencyPlot(Adult, support = 0.1, cex.names=0.8)
```



Usamos apriori para extraer los itemsets frecuentes con minsop 0.1. Orenamos por el valor de soporte. Inspeccionamos los 10 primeros

```
iAdult <- apriori(Adult, parameter = list(support = 0.1, target="frequent"))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##          NA    0.1    1 none FALSE            TRUE       5     0.1      1
##  maxlen            target   ext
##      10 frequent itemsets FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 4884
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[115 item(s), 48842 transaction(s)] done [0.03s].
## sorting and recoding items ... [31 item(s)] done [0.01s].
## creating transaction tree ... done [0.03s].
## checking subsets of size 1 2 3 4 5 6 7 8 9 done [0.10s].
## writing ... [2616 set(s)] done [0.00s].
## creating S4 object  ... done [0.02s].
```

```r
iAdult <- sort(iAdult, by="support")
inspect(head(iAdult, n=10))
```

```
##      items                                           support
## [1]  {capital-loss=None}                             0.9532779
## [2]  {capital-gain=None}                             0.9173867
## [3]  {native-country=United-States}                  0.8974243
## [4]  {capital-gain=None,capital-loss=None}           0.8706646
## [5]  {race=White}                                    0.8550428
## [6]  {capital-loss=None,native-country=United-States} 0.8548380
## [7]  {capital-gain=None,native-country=United-States} 0.8219565
## [8]  {race=White,capital-loss=None}                  0.8136849
## [9]  {race=White,native-country=United-States}       0.7881127
## [10] {race=White,capital-gain=None}                  0.7817862
```

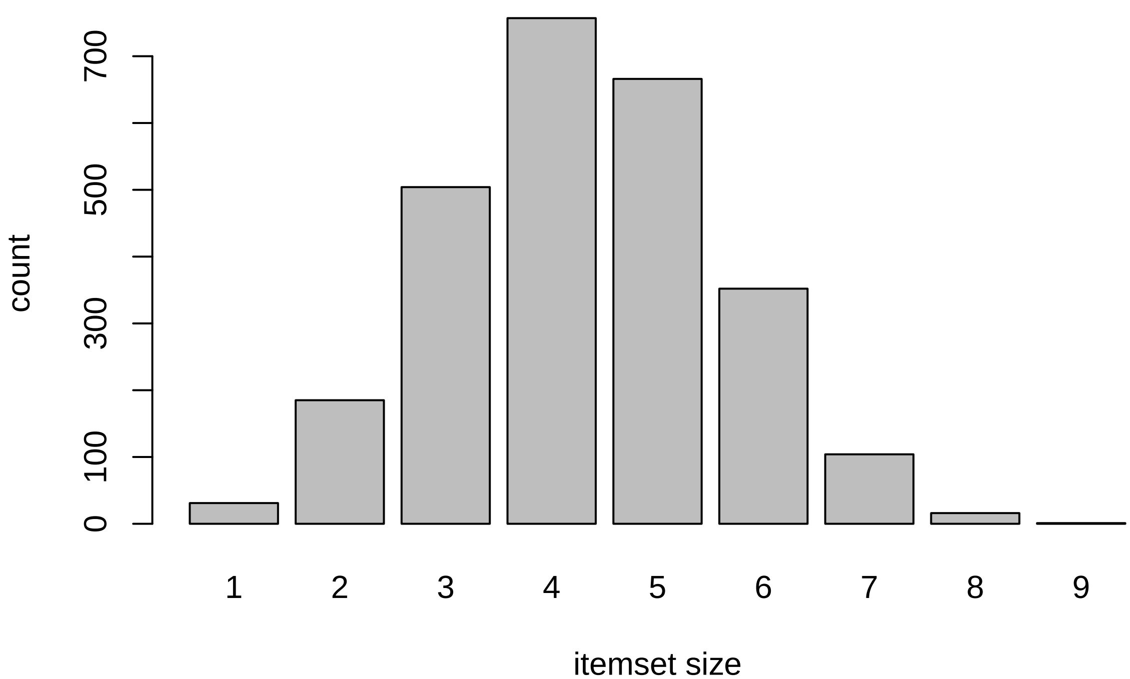
Podemos consultar el tamaño de los itemsets frecuentes (solo los 200 primeros)

```r
size(iAdult)[1:200]
```

```
##   [1] 1 1 1 2 1 2 2 2 2 2 3 3 3 3 1 4 1 2 2 2 2 3 2 2 2 3 2 1 3 3 3 3 2 3 3
##  [36] 2 3 3 4 3 3 4 3 4 2 4 4 1 1 4 4 3 2 4 2 2 3 2 5 3 2 3 1 4 5 2 2 2 3 3
##  [71] 3 3 3 3 2 3 4 2 2 2 4 4 3 3 3 2 2 2 3 4 3 3 1 2 2 3 3 2 3 3 4 4 3 3 4
## [106] 3 5 3 4 3 3 4 4 3 2 3 3 4 4 3 2 4 4 3 3 4 4 4 2 2 3 3 3 4 4 2 2 3 3 4
## [141] 3 3 3 5 4 2 3 3 3 4 4 4 3 3 2 3 3 5 5 4 5 4 3 3 4 4 4 5 4 3 4 4 5 3 4
## [176] 4 4 5 4 3 5 4 4 4 4 1 5 3 1 5 3 4 4 5 3 4 4 4 1 3
```

Representamos con un diagrama de barras

```r
barplot(table(size(iAdult)), xlab="itemset size", ylab="count")
```



Inspeccionamos los itemsets frecuentes de tamaño 1

```r
inspect(iAdult[size(iAdult)==1])
```

```
##      items                             support
## [1]  {capital-loss=None}               0.9532779
## [2]  {capital-gain=None}               0.9173867
## [3]  {native-country=United-States}    0.8974243
## [4]  {race=White}                      0.8550428
## [5]  {workclass=Private}               0.6941976
## [6]  {sex=Male}                        0.6684820
## [7]  {hours-per-week=Full-time}        0.5850907
## [8]  {income=small}                    0.5061218
## [9]  {age=Middle-aged}                 0.5051185
## [10] {marital-status=Married-civ-spouse} 0.4581917
## [11] {relationship=Husband}            0.4036690
## [12] {sex=Female}                      0.3315180
## [13] {marital-status=Never-married}    0.3299824
## [14] {education=HS-grad}               0.3231645
## [15] {age=Senior}                      0.2608616
## [16] {hours-per-week=Over-time}        0.2595307
## [17] {relationship=Not-in-family}      0.2576266
## [18] {education=Some-college}          0.2227182
## [19] {age=Young}                       0.1971050
## [20] {education=Bachelors}             0.1643053
## [21] {income=large}                    0.1605381
## [22] {relationship=Own-child}          0.1552148
## [23] {marital-status=Divorced}         0.1358052
## [24] {occupation=Prof-specialty}       0.1263667
## [25] {occupation=Craft-repair}         0.1251382
## [26] {occupation=Exec-managerial}      0.1246059
## [27] {hours-per-week=Part-time}        0.1210638
## [28] {occupation=Adm-clerical}         0.1148806
## [29] {occupation=Sales}                0.1126899
## [30] {relationship=Unmarried}          0.1049302
## [31] {occupation=Other-service}        0.1007944
```

Sacamos un vector lógico indicando que itemsets es máximal y mostramos los 6 primeros ordenados por su valor de soporte

```r
imaxAdult <- iAdult[is.maximal(iAdult)]
inspect(head(sort(imaxAdult, by="support")))
```

```
##     items                              support
## [1] {workclass=Private,
##      race=White,
##      sex=Male,
##      capital-gain=None,
##      capital-loss=None,
##      hours-per-week=Full-time,
##      native-country=United-States}      0.1774293
## [2] {workclass=Private,
##      race=White,
##      capital-gain=None,
```

```
##      capital-loss=None,
##      hours-per-week=Full-time,
##      native-country=United-States,
##      income=small}                         0.1578150
## [3] {workclass=Private,
##      race=White,
##      sex=Male,
##      capital-gain=None,
##      capital-loss=None,
##      native-country=United-States,
##      income=small}                         0.1560952
## [4] {age=Middle-aged,
##      workclass=Private,
##      race=White,
##      capital-gain=None,
##      capital-loss=None,
##      hours-per-week=Full-time,
##      native-country=United-States}         0.1456124
## [5] {marital-status=Married-civ-spouse,
##      relationship=Husband,
##      race=White,
##      sex=Male,
##      capital-gain=None,
##      capital-loss=None,
##      hours-per-week=Full-time,
##      native-country=United-States}         0.1429712
## [6] {race=White,
##      sex=Female,
##      capital-gain=None,
##      capital-loss=None,
##      native-country=United-States,
##      income=small}                         0.1339216
```

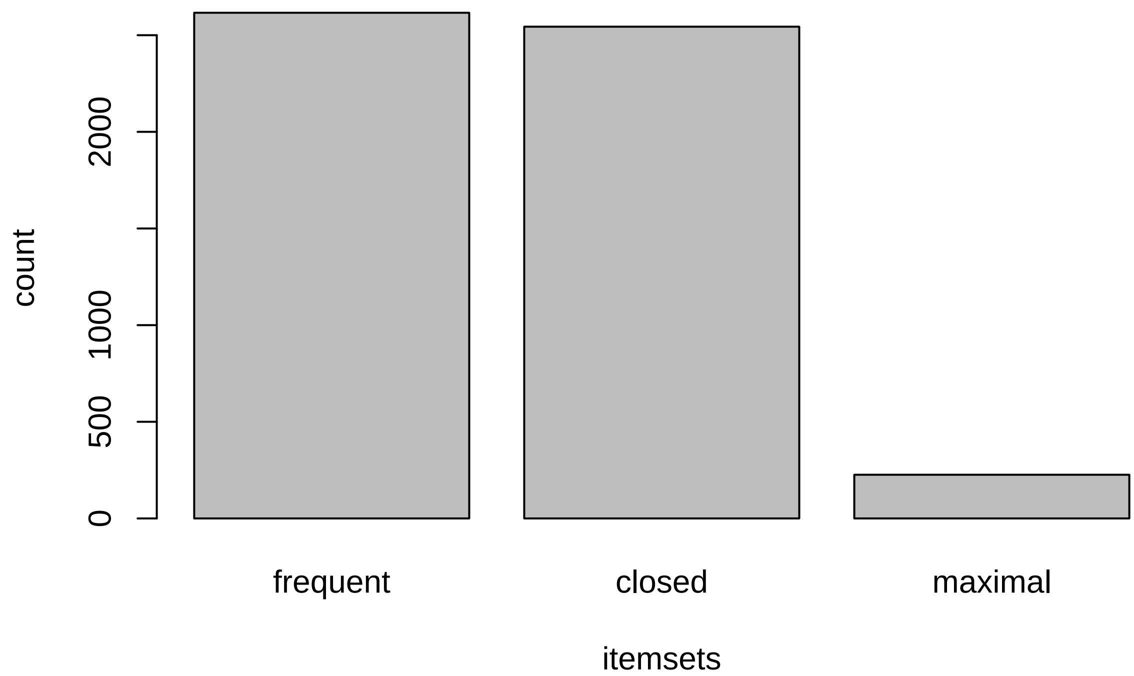
Sacamos un vector lógico indicando que itemsets es cerrado y mostramos los 6 primeros ordenados por su valor de soporte

```
icloAdult <- iAdult[is.closed(iAdult)]
inspect(head(sort(icloAdult, by="support")))
```

```
##     items                                          support
## [1] {capital-loss=None}                            0.9532779
## [2] {capital-gain=None}                            0.9173867
## [3] {native-country=United-States}                 0.8974243
## [4] {capital-gain=None,capital-loss=None}          0.8706646
## [5] {race=White}                                   0.8550428
## [6] {capital-loss=None,native-country=United-States} 0.8548380
```

Podemos pintar un gráfico de barras para ver la cantidad de itemsets frecuentes, cerrados y maximales que se han generado

```
barplot( c(frequent=length(iAdult), closed=length(icloAdult), maximal=length(imaxAdult)), ylab="count",
```



Usamos apriori para extraer las reglas con mínimo soporte 0.1 y confianza 0.8 con una longitud minima de 2. Obtenemos información resumida del conjunto

```r
rules <- apriori(Adult, parameter = list(support = 0.1, confidence = 0.8, minlen = 2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5     0.1      2
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 4884
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[115 item(s), 48842 transaction(s)] done [0.03s].
## sorting and recoding items ... [31 item(s)] done [0.01s].
## creating transaction tree ... done [0.02s].
## checking subsets of size 1 2 3 4 5 6 7 8 9 done [0.09s].
## writing ... [6133 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```r
summary(rules)
```

```
## set of 6133 rules
##
```

```
## rule length distribution (lhs + rhs):sizes
##    2    3    4    5    6    7    8    9
##  121  637 1510 1903 1345  511   99    7
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   4.000   5.000   4.926   6.000   9.000
##
## summary of quality measures:
##     support         confidence          lift
##  Min.   :0.1000   Min.   :0.8004   Min.   :0.9169
##  1st Qu.:0.1158   1st Qu.:0.8895   1st Qu.:0.9911
##  Median :0.1353   Median :0.9241   Median :1.0197
##  Mean   :0.1700   Mean   :0.9236   Mean   :1.2044
##  3rd Qu.:0.1890   3rd Qu.:0.9587   3rd Qu.:1.0783
##  Max.   :0.8707   Max.   :1.0000   Max.   :2.9421
##
## mining info:
##   data ntransactions support confidence
##  Adult         48842     0.1        0.8
```

Podemos ver las reglas (lhs es el antecedente y rhs el consecuente de la regla) y sus valores para las medidas soporte, confianza y lift. También podemos ver solo los valores de las medidas de calidad

```r
inspect(head(rules))
```

```
##     lhs                            rhs                             support confidence      lift
## [1] {relationship=Unmarried}  => {capital-loss=None}             0.1019819  0.9719024 1.0195373
## [2] {occupation=Sales}        => {race=White}                    0.1005282  0.8920785 1.0433144
## [3] {occupation=Sales}        => {native-country=United-States} 0.1039679  0.9226017 1.0280552
## [4] {occupation=Sales}        => {capital-gain=None}             0.1030670  0.9146076 0.9969706
## [5] {occupation=Sales}        => {capital-loss=None}             0.1068343  0.9480378 0.9945030
## [6] {occupation=Adm-clerical} => {native-country=United-States} 0.1052373  0.9160577 1.0207632
```

```r
quality(head(rules))
```

```
##     support confidence      lift
## 1 0.1019819  0.9719024 1.0195373
## 2 0.1005282  0.8920785 1.0433144
## 3 0.1039679  0.9226017 1.0280552
## 4 0.1030670  0.9146076 0.9969706
## 5 0.1068343  0.9480378 0.9945030
## 6 0.1052373  0.9160577 1.0207632
```

Podemos ordenar las reglas por el campo que más nos interese

```r
rulesSorted = sort(rules, by="confidence")
inspect(head(rulesSorted))
```

```
##     lhs                                      rhs          support confidence     lift
## [1] {relationship=Husband,
##      income=large}                        => {sex=Male} 0.1211662          1 1.495926
```

```
## [2] {relationship=Husband,
##      hours-per-week=Over-time}          => {sex=Male} 0.1472298          1 1.495926
## [3] {age=Senior,
##      relationship=Husband}              => {sex=Male} 0.1479874          1 1.495926
## [4] {marital-status=Married-civ-spouse,
##      relationship=Husband,
##      income=large}                      => {sex=Male} 0.1210843          1 1.495926
## [5] {relationship=Husband,
##      race=White,
##      income=large}                      => {sex=Male} 0.1111339          1 1.495926
## [6] {relationship=Husband,
##      native-country=United-States,
##      income=large}                      => {sex=Male} 0.1110724          1 1.495926
```

Seleccionar un subconjunto de reglas que cumplan una condición. Por ejemplo, seleccionamos las reglas que tenga lift > 1.2 y que en el consecuente de la regla tengan el itemset race=White

```r
rulesRaceWhite <- subset(rules, subset = lhs %in% "race=White" & lift > 1.2)
inspect(head(rulesRaceWhite))
```

```
##     lhs                              rhs                                  support confidence     lift
## [1] {occupation=Craft-repair,
##      race=White}                  => {sex=Male}                         0.1076532  0.9553052 1.429066
## [2] {relationship=Own-child,
##      race=White}                  => {marital-status=Never-married}     0.1168257  0.8959020 2.714999
## [3] {race=White,
##      income=large}                => {marital-status=Married-civ-spouse} 0.1249949  0.8578053 1.872154
## [4] {race=White,
##      income=large}                => {sex=Male}                         0.1246673  0.8555571 1.279851
## [5] {age=Young,
##      race=White}                  => {marital-status=Never-married}     0.1440154  0.8512647 2.579728
## [6] {race=White,
##      hours-per-week=Over-time} => {sex=Male}                            0.1956103  0.8239759 1.232607
```

Eliminar las reglas redundantes

```r
subsetMatrix <- is.subset(rulesSorted, rulesSorted)
subsetMatrix[lower.tri(subsetMatrix, diag=T)] <- NA
redundant <- colSums(subsetMatrix, na.rm=T) >= 1
rulesPruned <- rulesSorted[!redundant] # remove redundant rules
inspect(head(rulesPruned))
```

```
##     lhs                              rhs                                  support confidence     lift
## [1] {relationship=Husband,
##      income=large}                => {sex=Male}                         0.1211662  1.0000000 1.495926
## [2] {relationship=Husband,
##      hours-per-week=Over-time} => {sex=Male}                            0.1472298  1.0000000 1.495926
## [3] {age=Senior,
##      relationship=Husband}     => {sex=Male}                            0.1479874  1.0000000 1.495926
## [4] {relationship=Husband}     => {sex=Male}                            0.4036485  0.9999493 1.495851
## [5] {age=Senior,
##      relationship=Husband}     => {marital-status=Married-civ-spouse} 0.1479669  0.9998616 2.182191
```

```
## [6] {relationship=Husband,
##      race=White,
##      hours-per-week=Full-time} => {marital-status=Married-civ-spouse} 0.1886491  0.9996745 2.181782
```

También podemos calcular para itemsets o para reglas otras medidas de calidad. Podemos calcular estas medidas para nuestras reglas podadas y añadirselas a la sección quality para que los valores de las medidas nuevas salgan también cuando inspeccionamos las reglas:

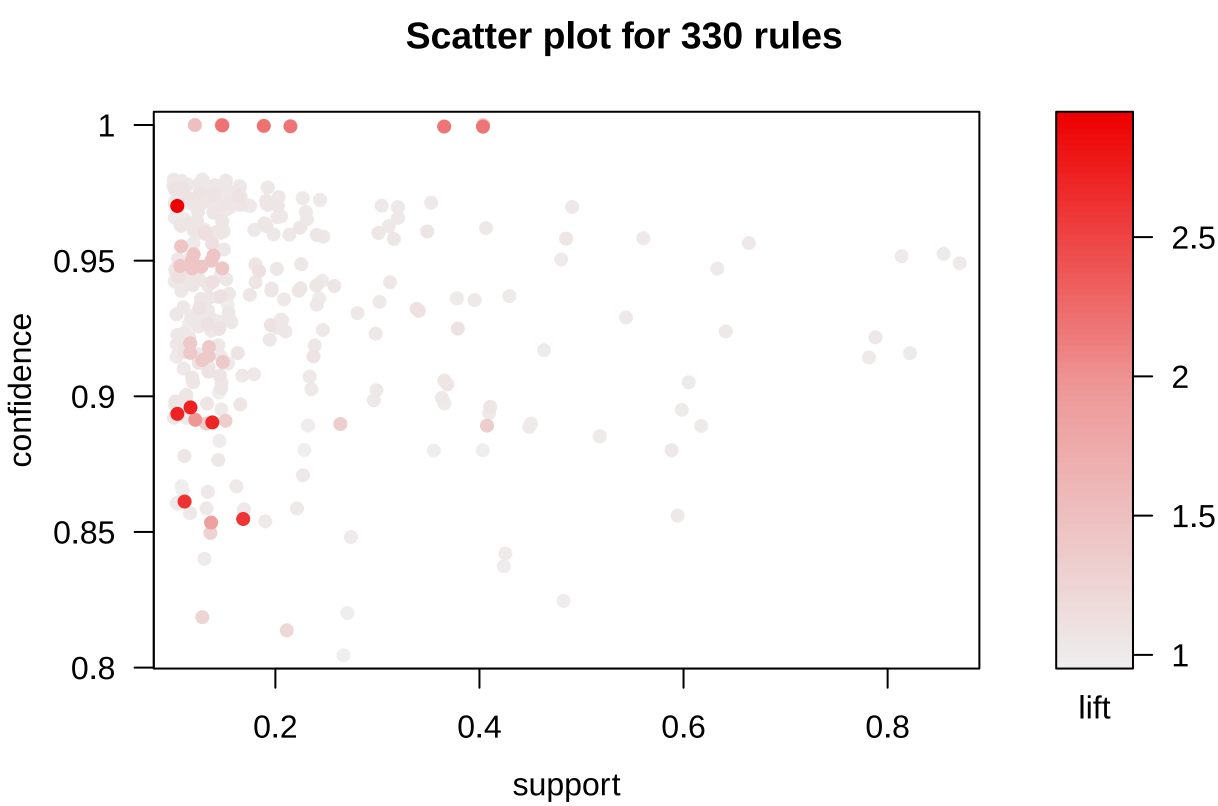
```r
mInteres <- interestMeasure(rulesPruned, measure=c("hyperConfidence", "leverage" ,"phi", "gini"), transa
quality(rulesPruned) <- cbind(quality(rulesPruned), mInteres)
inspect(head(sort(rulesPruned, by="phi")))
```

```
##     lhs                            rhs                                support confidence     lift l
## [1] {relationship=Husband}      => {marital-status=Married-civ-spouse} 0.4034233  0.9993914 2.181164
## [2] {relationship=Husband,
##      race=White}                => {marital-status=Married-civ-spouse} 0.3654232  0.9994400 2.181270
## [3] {relationship=Husband}      => {sex=Male}                          0.4036485  0.9999493 1.495851
## [4] {relationship=Husband,
##      hours-per-week=Full-time} => {marital-status=Married-civ-spouse} 0.2147742  0.9995236 2.181453
## [5] {age=Young}                 => {marital-status=Never-married}      0.1684820  0.8547834 2.590391
## [6] {relationship=Husband,
##      race=White,
##      hours-per-week=Full-time} => {marital-status=Married-civ-spouse} 0.1886491  0.9996745 2.181782
```

```r
library(arulesViz)
```

Utilizar la función plot para representar las reglas en función de las medidas de calidad

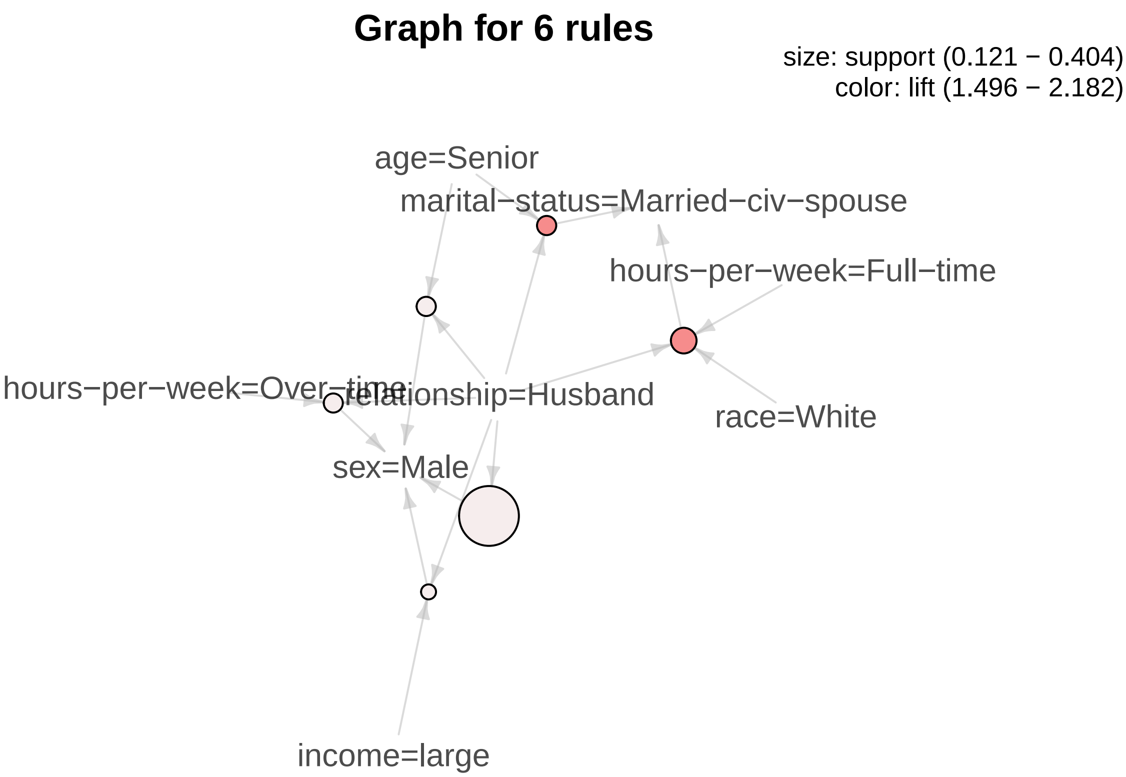
```r
plot(rulesPruned)
```



Podemos modificar el tipo de gráfico generado cambiando el parámetro método de la función plot. Además, se puede modificar el gráfico cambiando los parámetros del tipo de gráfico

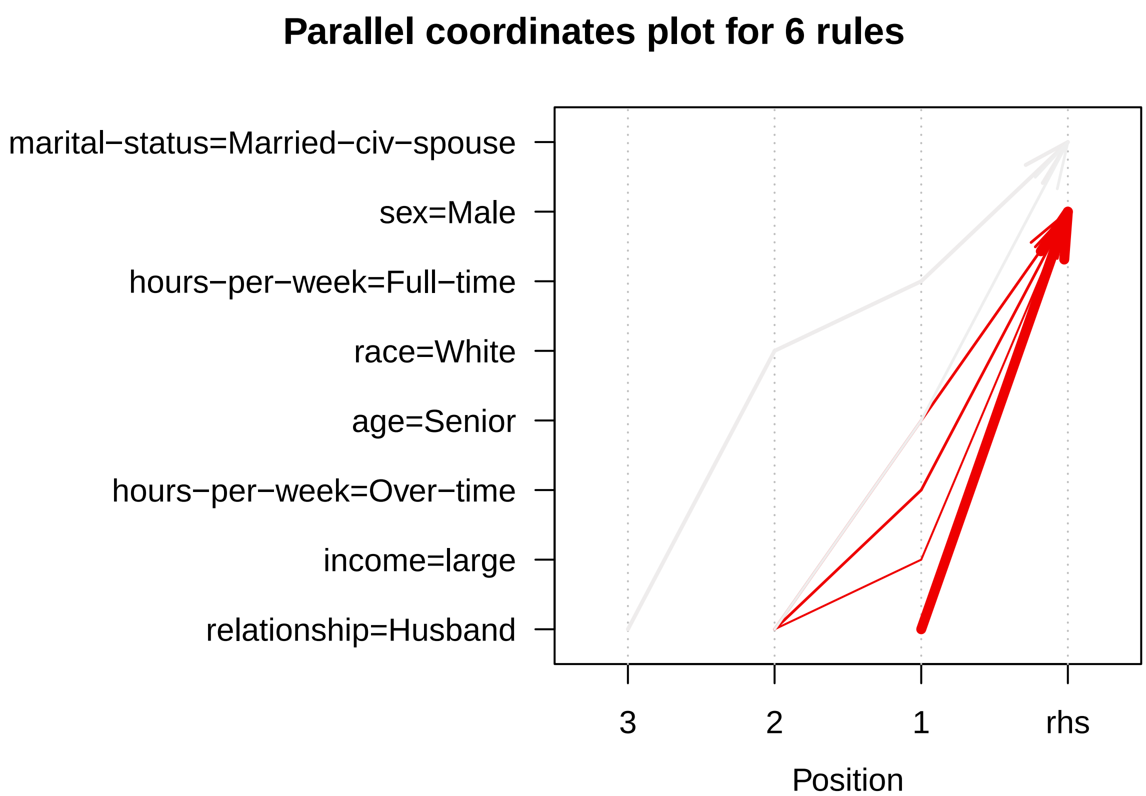
```
??plot # consultar las distintas opciones para la función plot
```

```
plot(rulesPruned[1:6], method="graph", control=list(type="items"))
```



Podemos visualizar las reglas como una matriz agrupada. Los antecedentes en las columnas son agrupados usando clustering. En modo interactivo podemos hacer zoom del nodo que queramos estudiar y acceder a las reglas que lo componen para inspeccionarlas.

```
#try: plot(rulesPruned, method="grouped", interactive=TRUE)
plot(rulesPruned[1:6], , method="paracoord", control=list(reorder=TRUE))
```



Las podemos guardar en texto plano usando la función write. En este ejemplo las guardamos en un fichero llamado data.csv, usamos como separador "," y no le ponemos ningún nombre a las columnas

```
write(rulesPruned, file="reglas.csv", sep = ",", col.names=NA)
```

También las podemos guardar en formato PMML. Así podemos volver a leerlas

```
library(pmml)
```

```
## Loading required package: XML

## Warning: package 'XML' was built under R version 3.3.2
```

```
write.PMML(rulesPruned,file="reglas.pmml")
```

```
## [1] "reglas.pmml"
```

```
reglasPMML = read.PMML("reglas.pmml")
```