

Extracción de reglas

Laura del Pino Díaz

23/1/2017

Índice

Introducción	2
Preparación de los datos	3
Visualización de las transacciones	5
Itemset frecuentes	6
Comparación del número de itemset frecuentes, conjuntos maximales y conjuntos cerrados	8
Obtención de reglas	9

Introducción

La extracción de reglas de asociación es una técnica de la minería de datos que nos permite extraer conocimiento de las bases de datos. En este trabajo trata de extraer conocimiento de la base de datos *mushrooms*.

La base de datos *mushrooms* contiene los atributos que describen las setas y que permitirían clasificar las setas observadas en venenosas y comestibles. Los atributos se listan a continuación:

- cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
- cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
- cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r,pink=p,purple=u,red=e,white=w,yellow=y
- bruises: bruises=t,no=f
- odor: almond=a,anise=l,creosote=c,fishy=y,foul=f,musty=m,none=n,pungent=p,spicy=s
- gill-attachment: attached=a,descending=d,free=f,notched=n
- gill-spacing: close=c,crowded=w,distant=d
- gill-size: broad=b,narrow=n
- gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g, green=r,orange=o, pink=p,purple=u,red=e,white=w,yellow=y
- stalk-shape: enlarging=e,tapering=t
- stalk-root: bulbous=b,club=c,cup=u,equal=e,rhizomorphs=z,rooted=r,missing=?
- stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s
- stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s
- stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y
- stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y
- veil-type: partial=p,universal=u
- veil-color: brown=n,orange=o,white=w,yellow=y
- ring-number: none=n,one=o,two=t
- ring-type: cobwebby=c,evanescent=e,flaring=f,large=l,none=n,pendant=p,sheathing=s,zone=z
- spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r,orange=o,purple=u,white=w,yellow=y
- population: abundant=a,clustered=c,numerous=n,scattered=s,several=v,solitary=y
- habitat: grasses=g,leaves=l,meadows=m,paths=p,urban=u,waste=w,woods=d

```
datos = read.csv("mushrooms.csv")
```

Preparación de los datos

Para realizar la extracción de reglas tenemos que preparar los datos para que estén en un formato que nos facilite el trabajo. Este formato es un formato binario donde los valores sean 0 y 1 que nos permita extraer el los casos en los que aparece el atributo y en los que no, para métricas como la *confianza confirmada* es necesario conocer los casos en donde se da un valor y su contrario.

Esta transformación la conseguimos con el siguiente conjunto de funciones:

```
transformDataToBinary <- function(lvl, data) {
  result = sapply(data, function(x) {
    ifelse(x == lvl, 1, 0)
  })
  return(result)
}

selectData <- function(column) {
  if (is.factor(column) && length(levels(column)) > 2) {
    lvls = levels(column)
    m = matrix(nrow = 8124, ncol = 0)
    m = data.frame(m)
    for (lvl in lvls) {
      d = transformDataToBinary(lvl, column)
      d = as.factor(d)
      m = cbind(m, d)
    }
    colnames(m) <- lvls
    return(m)
  } else if (!is.factor(column)) {
    return(selectData(as.factor(column)))
  } else {
    return(column)
  }
}

expandedDataFrame = cbind(datos[, -19], selectData(datos[, "ring.number"]))
```

En estos momentos en *expandedDataFrame* tenemos 25 variables, las 22 anteriores y las tres variables binarias creadas a partir de la variable *ring.number*.

Por suerte todas las variables son categóricas, lo que nos facilita el preprocesamiento al no tener que dividir ningún dominio en intervalos.

Ahora que ya tenemos el dataframe en un estado que nos favorece nos queda que cambiar la interpretación que le damos, para que sea del tipo transacción. Para ello cargamos el paquete *arules* que nos permitirá hacer dicha transformación y además cargaremos el paquete de visualización de reglas de asociación *arulesViz* que será útil en pasos posteriores.

```
require(arules)
require(arulesViz)

transactionData <- as(expandedDataFrame, "transactions")
summary(transactionData)
```

transactions as itemMatrix in sparse format with
 8124 rows (elements/itemsets/transactions) and
 122 columns (items) and a density of 0.204918

most frequent items:

veil.type=p	n=0	veil.color=w
8124	8088	7924
gill.attachment=f	t=0	(Other)
7914	7524	163526

element (itemset/transaction) length distribution:

sizes

25
 8124

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
25	25	25	25	25	25

includes extended item information - examples:

	labels	variables	levels
1	class=e	class	e
2	class=p	class	p
3	cap.shape=b	cap.shape	b

includes extended transaction information - examples:

	transactionID
1	1
2	2
3	3

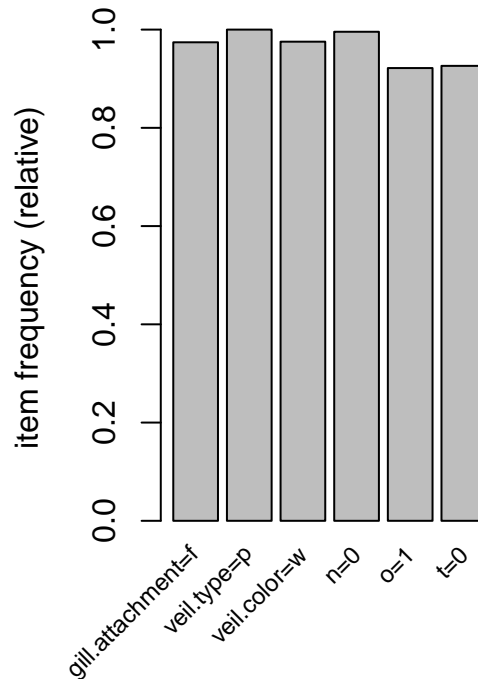
Del resumen anterior obtenermos que los elementos más repetidos son *veil.type=p*, que se da para todas las observaciones de la base de datos, seguido por *n=0* que es una de las variables generadas a partir de *ring.number* de carácter binaria, por lo que se estima que sea más normal que las otras dos variables tomen valores, en concreto la variable *o* dado que la variable *t* aparece como item frecuente con el valor 0. A parte de estas tres variables tenemos como item frecuentes *veil.color* con el valor *w* y *gill.attachment* con el valor *f*

Además obtenemos la información sobre la longitud máxima de la transacción que en todos los casos es 25, el número de variables, esto es debido a que no falta ninguna variable.

Visualización de las transacciones

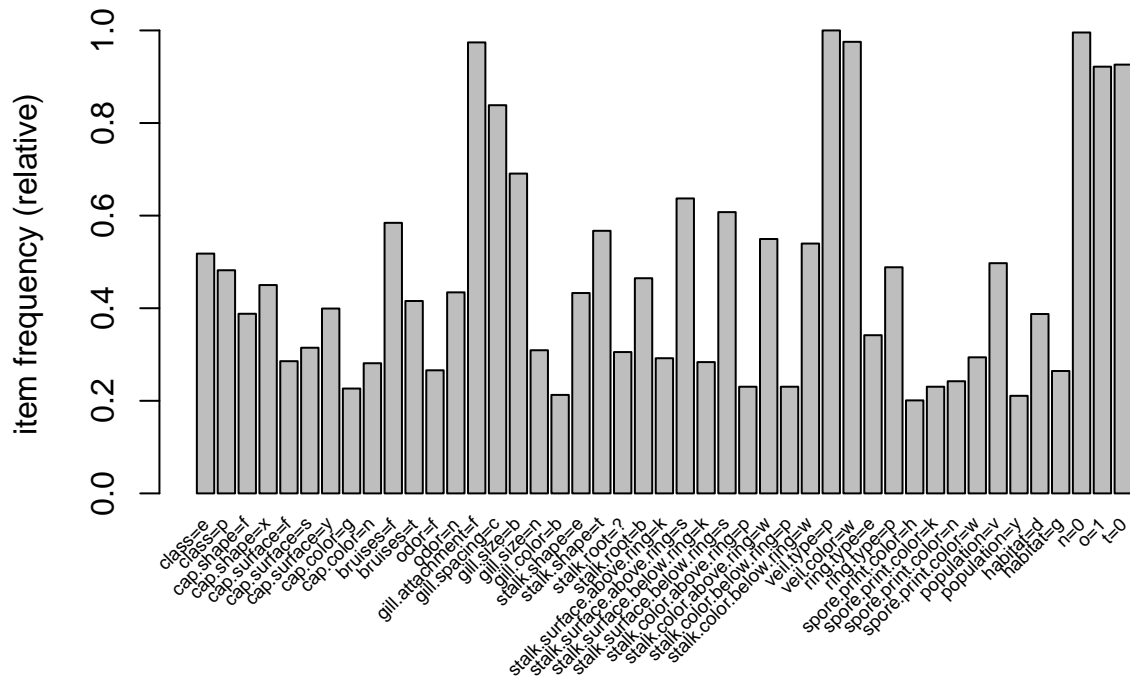
En el caso de que las variables tomaran solamente los valores *si* o *no* podríamos y que el número de variables en cada transacción fuese distinto podría ser interesante realizar una visualización de las transacciones. Pero dado que este no es el caso solamente visualizaremos los conjuntos de items más frecuentes, como en la siguiente imagen, donde miramos aquellos items que tienen un valor de soporte mayor que 0.9.

```
itemFrequencyPlot(transactionData, support = 0.9, cex.names = 0.75)
```



De la imagen anterior podemos deducir que aquellas reglas con las variables *gill.attachment=f*, *veil.type=p*, *veil.color=w*, *n=0*, *o=1* y *t=0* serán reglas que no aportarán información puesto que se cumplirán en al menos el 90% de los casos recogidos en la base de datos.

```
itemFrequencyPlot(transactionData, support = 0.2, cex.names = 0.6)
```



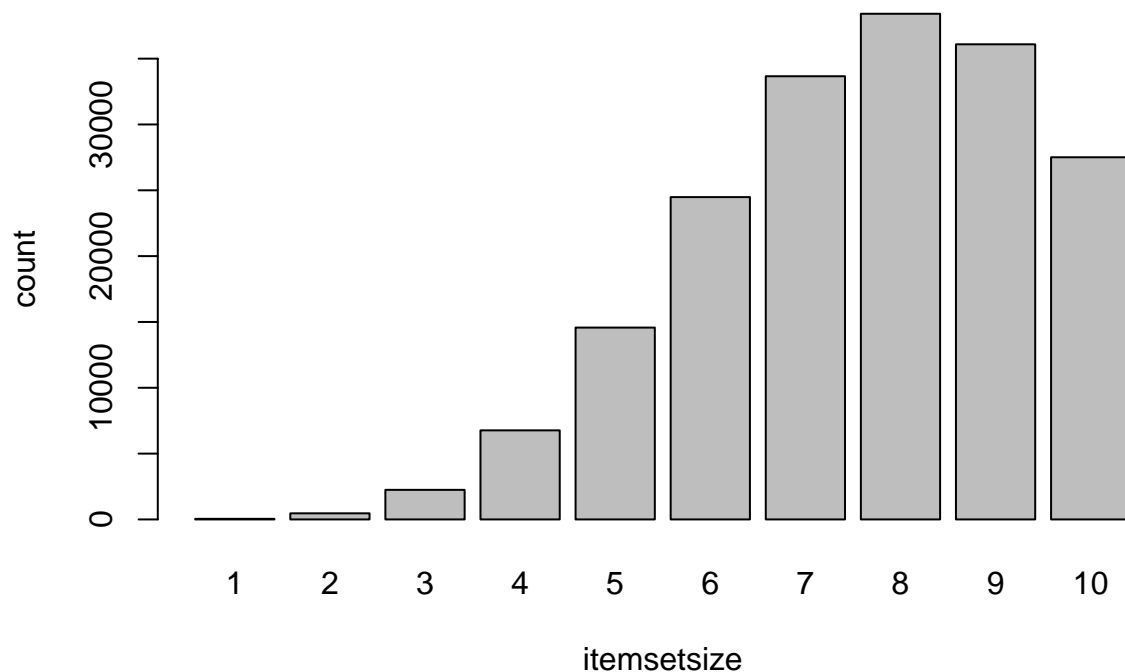
El gráfico anterior representa una población de aproximadamente 40 ítems que se convierten en frecuentes al determinar el umbral de soporte en 0.2 . 40 ítems representa una cantidad significativa de los valores que pueden tomar las variables de nuestra base de datos. Podemos dejar el soporte mínimo en 0.2, para tener un conjunto significativo de ítems con los que trabajar.

Itemset frecuentes

Para extraer los itemset frecuentes, utilizaremos el método *apriori* que irá buscando para cada longitud de itemset aquellos itemsets que tengan un valor de soporte mayor que el que se ha determinado como mínimo, en nuestro caso 0.2. Tras ello lo ordenaremos por soporte y mostraremos una relación entre la longitud del itemset y la frecuencia absoluta de cada longitud. Siempre teniendo en cuenta que se nos van a devolver solamente los itemset de longitud menor o igual a 10.

```
aPrioriTransactionData = apriori(transactionData, parameter = list(support = 0.2,
  target = "frequent"), control = list(verbose = F))
aPrioriTransactionData = sort(aPrioriTransactionData, by = "support")
```

```
barplot(table(size(aPrioriTransactionData)), xlab = "itemsetsize",
          ylab = "count")
```



En la gráfica anterior podemos ver como aumenta el número de superconjuntos con la longitud del itemset, manteniéndose estos nuevos conjuntos como frecuentes siguiendo la propiedad antimonótona de los itemset que afirma que cualquier subconjunto de un itemset tiene mayor o igual soporte que el itemset del que proviene.

Vamos a inspeccionar el soporte de los itemsets de longitud 1 que hemos obtenido.

```
inspect(head(aPrioriTransactionData[size(aPrioriTransactionData) ==
1], 10))
```

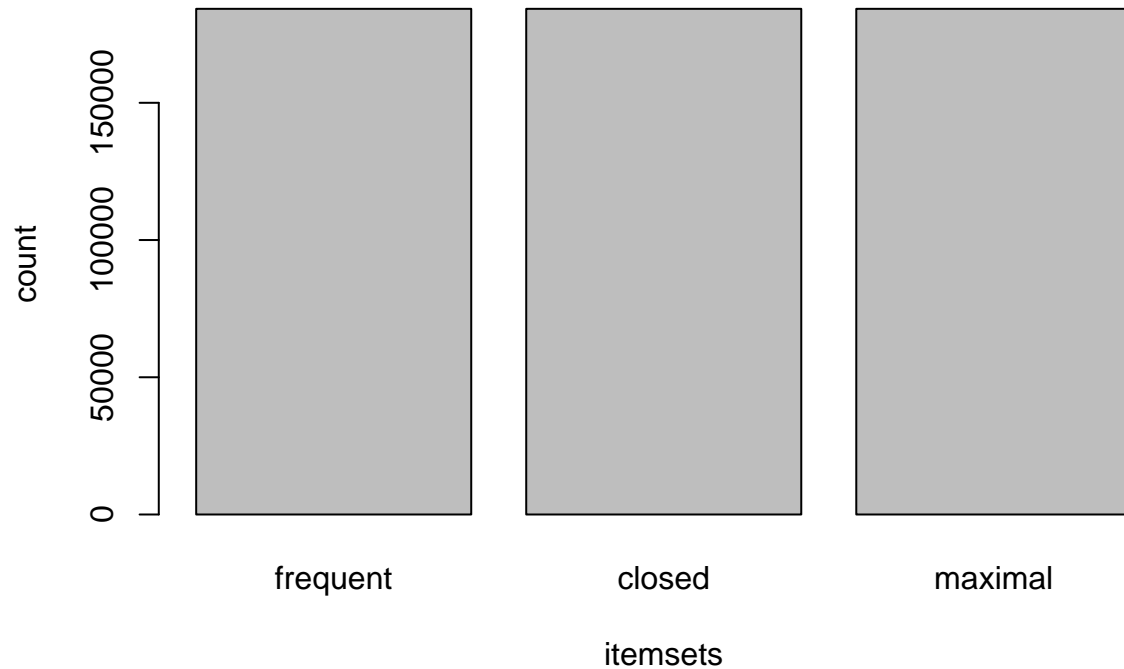
	items	support
[1]	{veil.type=p}	1.0000000
[2]	{n=0}	0.9955687
[3]	{veil.color=w}	0.9753816
[4]	{gill.attachment=f}	0.9741507
[5]	{t=0}	0.9261448
[6]	{o=1}	0.9217134
[7]	{gill.spacing=c}	0.8385032
[8]	{gill.size=b}	0.6907927
[9]	{stalk.surface.above.ring=s}	0.6371246
[10]	{stalk.surface.below.ring=s}	0.6075825

Los primeros 4 elementos tienen un soporte realmente alto, por lo que cualquier regla que se produzca con ellos puede no dar información relevante.

Comparación del número de itemset frecuentes, conjuntos maximales y conjuntos cerrados

Para conocer mejor los itemset obtenidos por el método *a priori* realizaremos una comparativa de los conjuntos maximales, cerrados y frecuentes.

```
iclosedAPrioriTransactionData <- aPrioriTransactionData[is.closed(aPrioriTransactionData)]
imaxAPrioriTransactionData <- aPrioriTransactionData[is.maximal(aPrioriTransactionData)]
barplot(c(frequent = length(aPrioriTransactionData), closed = length(aPrioriTransactionData),
         maximal = length(aPrioriTransactionData)), ylab = "count",
        xlab = "itemsets")
```



Al ver el número de itemset cerrados y maximales tan igual, podemos decir que los conjuntos son iguales, por lo que podemos prescindir del conjunto de los maximales, dado que el conjunto de los itemset cerrados nos permite tener la lista de los más frecuentes y a la vez tener la información del soporte.

Obtención de reglas

Para la obtención de reglas usamos `apriori` y le indicamos que el soporte mínimo es de 0.2 como ya habíamos establecido antes. Además le indicamos que la confianza mínima de la regla es de 0.75, este valor lo establecemos así para tener mayor conjunto de reglas que estudiar aparte de las que van a tener un valor alto por contener aquellos items que tienen soporte muy próximo a 1.

```
rules <- apriori(transactionData, parameter = list(support = 0.2,
  confidence = 0.75, minlen = 2), control = list(verbose = F))

summary(rules)
```

set of 1323192 rules

```
rule length distribution (lhs + rhs):sizes
  2      3      4      5      6      7      8      9
367    3843   18900   58360  129447  220723  297860  320281
10
273411
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
	2.000	7.000	8.000	8.087	9.000	10.000

summary of quality measures:

	support	confidence	lift
Min.	:0.2001	Min. :0.7500	Min. :0.8232
1st Qu.:	0.2127	1st Qu.:1.0000	1st Qu.:1.0265
Median :	0.2127	Median :1.0000	Median :1.4476
Mean :	0.2233	Mean :0.9842	Mean :1.6706
3rd Qu.:	0.2166	3rd Qu.:1.0000	3rd Qu.:2.0474
Max. :	0.9956	Max. :1.0000	Max. :4.7014

mining info:

	data	ntransactions	support	confidence
transactionData		8124	0.2	0.75

Dado que el conjunto de reglas es muy amplio vamos a hacer una visualización de las primeras reglas:

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

	lhs	rhs	support	confidence	lift
[1]	{spore.print.color=h}	=> {gill.spacing=c}	0.2008863	1	1.192601
[2]	{spore.print.color=h}	=> {o=1}	0.2008863	1	1.084936
[3]	{spore.print.color=h}	=> {t=0}	0.2008863	1	1.079745
[4]	{spore.print.color=h}	=> {gill.attachment=f}	0.2008863	1	1.026535
[5]	{spore.print.color=h}	=> {veil.color=w}	0.2008863	1	1.025240
[6]	{spore.print.color=h}	=> {n=0}	0.2008863	1	1.004451

Como podemos ver, aparecen a la cabeza aquellas reglas que tienen un itemset con un soporte alto en el consecuente, por lo que da igual lo que se ponga en el antecedente, la regla tendrá una confianza alta. Este tipo de reglas las eliminamos con el siguiente código.

```
rulesConfidenceInteresting <- subset(rules, subset = confidence <=
  0.9)
inspect(head(rulesConfidenceInteresting))
```

	lhs	rhs	support	confidence	lift
[1]	{spore.print.color=k}	=> {class=e}	0.2028557	0.8803419	1.699595
[2]	{spore.print.color=k}	=> {stalk.surface.above.ring=s}	0.2067947	0.8974359	1.408572
[3]	{spore.print.color=n}	=> {class=e}	0.2146726	0.8861789	1.710864
[4]	{spore.print.color=n}	=> {stalk.surface.below.ring=s}	0.2067947	0.8536585	1.405008
[5]	{spore.print.color=n}	=> {gill.size=b}	0.2028557	0.8373984	1.212228
[6]	{habitat=g}	=> {stalk.color.below.ring=w}	0.2112260	0.7988827	1.480411

Para extraer información de valor vamos a focalizar el estudio de las reglas en unas pocas, para seleccionar las más interesantes lo primero que haremos es mostrar las reglas en una comparación entre su soporte y su confianza.

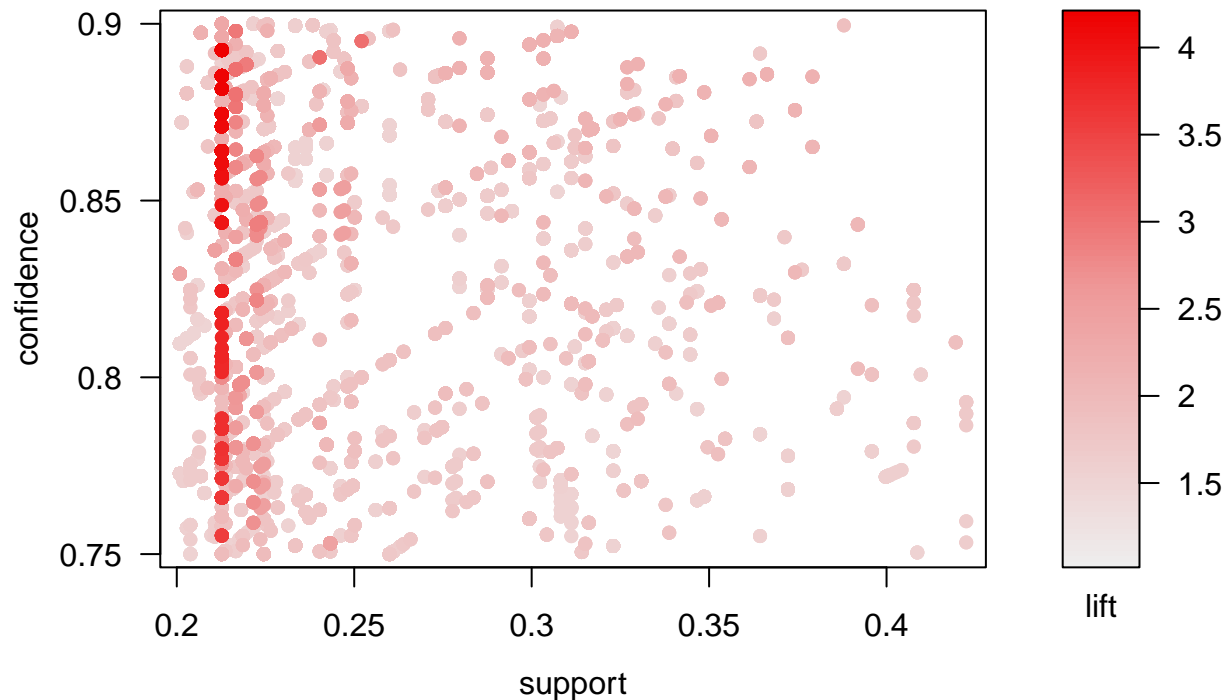
```
plot(rulesConfidenceInteresting)
```



Aunque es tentador seleccionar aquellas reglas que tienen mayor soporte y confianza, situadas a la derecha en la gráfica, hay que tener el valor de lift con el que se colorea la regla. En los casos que maximizan la confianza y el soporte tienen un valor de lift próximo a 1, este valor 1 significa independencia estadística por lo que los items que se relacionan en estos casos no tienen una relación de causalidad. Es por esto que hacemos otro refinamiento de las reglas obtenidas para dejar fuera este conjunto y nos quedamos con las reglas con lift mayor a 1.5.

```
rulesInteresting <- subset(rulesConfidenceInteresting, subset = lift >
  1.5)
plot(rulesInteresting)
```

Scatter plot for 46950 rules



El gráfico anterior nos muestra que refinar la selección para evitar aquellas reglas que tienen independencia estadística nos ha supuesto quitar casi 30 000 reglas del estudio.

Podemos servirnos de que tenemos tres variables que toman los valores “sí o no” representados como 1 y 0 respectivamente para calcular otra métrica: *la confianza confirmada*. Esta métrica se define como la confianza de que dado X suceda Y menos el valor de que dado X suceda no-Y.

```
interestingMeasure <- interestMeasure(rulesInteresting, measure = "confirmedConfidence",
  transactions = transactionData)
quality(rulesInteresting) <- cbind(quality(rulesInteresting),
  interestingMeasure)
inspect(head(sort(rulesInteresting, by = "interestingMeasure")))
```

	lhs	rhs	support	confidence	lift	interestingMeasure
[1]	{odor=n, gill.size=b, ring.type=p, o=1}	=> {habitat=d}	0.2127031	0.9	2.322618	0.8
[2]	{odor=n, gill.size=b, ring.type=p, t=0}	=> {habitat=d}	0.2127031	0.9	2.322618	0.8
[3]	{odor=n, stalk.shape=t, stalk.surface.above.ring=s, stalk.surface.below.ring=s}	=> {habitat=d}	0.2127031	0.9	2.322618	0.8

[4] {odor=n, gill.spacing=c, gill.size=b, o=1}	=> {habitat=d} 0.2127031	0.9 2.322618	0.8
[5] {odor=n, gill.spacing=c, gill.size=b, t=0}	=> {habitat=d} 0.2127031	0.9 2.322618	0.8
[6] {odor=n, gill.size=b, ring.type=p, o=1}	=> {bruises=t} 0.2127031	0.9 2.165758	0.8